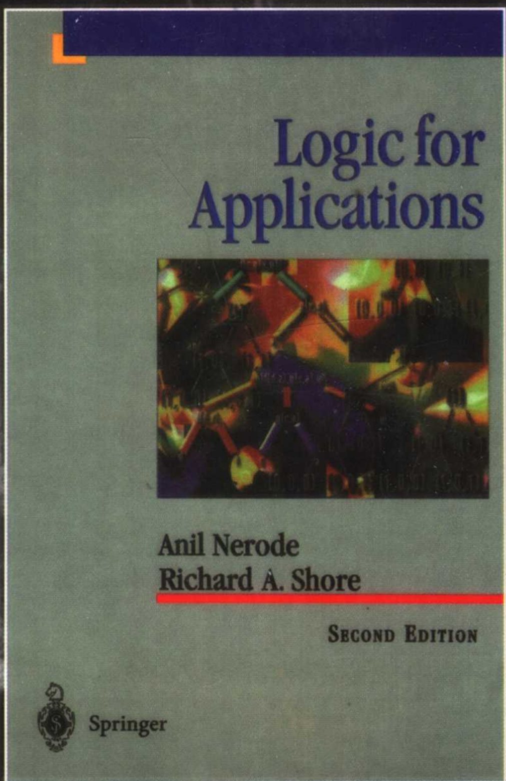
计 算 机 科 学 丛 书

原书第2版

应用逻辑

(美) Anil Nerode Richard A. Shore 著 丁德成 徐亚涛 吴永成 金陈园 译
康奈尔大学 南京大学



Logic for Applications

Second Edition

机械工业出版社
China Machine Press

应用逻辑（原书第2版）

“本书无疑是计算机科学最富成效的入门教科书之一……我们强烈建议把它作为教科书……”

——美国计算机协会自动机与可计算性理论专业组（SIGACT）

这是一本结合逻辑在计算机科学中的应用来介绍数理逻辑的教科书。书中主要介绍了消解定理证明、逻辑式程序设计和非经典逻辑（模态逻辑和直觉主义逻辑），所用的方法与数理逻辑的经典著作有很大不同，更加适合研究计算机理论的读者，可以帮助他们更好地理解计算机理论中的许多概念，是一本真正面向计算机科学的逻辑著作。另外，每章最后给出了进一步阅读建议，书末又分主题给出了相当多的参考文献，便于读者深入学习。

本书不要求读者具备逻辑基础知识，适合计算机科学系和数学系高年级本科生以及低年级研究生使用。

作者简介

Anil Nerode 康奈尔大学数学系的创始人和教授，于1956年在芝加哥大学获得博士学位。他的研究领域包括数理逻辑、自动机、可计算理论、混合系统等。除本书外，他还与其他人合著了《Effective Completeness Theorems for Modal Logic》、《Tableaux for Constructive Concurrent Dynamic Logic》、《Logic, Categories, Lambda Calculus》等书。

Richard A. Shore 康奈尔大学数学教授，于1972年在麻省理工学院获得博士学位。他的研究领域包括数理逻辑、递归论、集合论等。

影印版

ISBN 7-111-19772-0

定价：49.00元

上架指导：计算机／数理逻辑

ISBN 978-7-111-21404-5

9 787111 214045

ISBN 978-7-111-21404-5

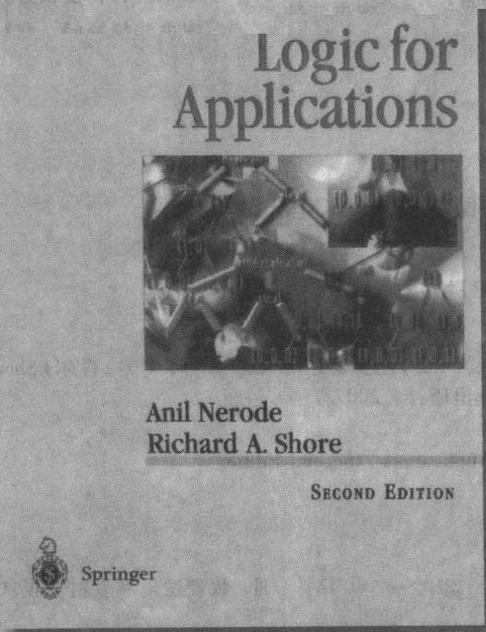
定价：38.00 元

封面设计·锡 彬

计 算 机 科 学 丛 书

应用逻辑

（美）Anil Nerode　Richard A. Shore 著　丁德成 徐亚涛 吴永成 金陈园 译
康奈尔大学　南京大学



Logic for Applications
Second Edition

机械工业出版社
China Machine Press

本书是介绍数理逻辑的基础教材，不仅覆盖了传统的基本内容(语法、语义、可靠性、完全性和紧致性)，而且很大一部分是讨论非传统的内容，诸如消解定理证明、逻辑式程序设计和非经典逻辑（模态逻辑和直觉主义逻辑），而这些主题在现代计算机科学中变得越来越重要。

本书讲述的内容广泛，深入浅出，简明易懂，适合作为计算机科学系和数学系高年级本科生以及低年级研究生的教材。

Translation from the English language edition: *Logic for Applications*, *Second Edition* (ISBN 0-387-94893-7) by Anil Nerode and Richard A. Shore.

本书版权登记号：图字：01-2006-3887

图书在版编目(CIP)数据

应用逻辑(原书第2版)/(美)尼罗德(Nerode, A.)，(美)肖尔(Shore, R. A.)著；丁德成等译. —北京：机械工业出版社，2007.7

(计算机科学丛书)

书名原文：Logic for Applications, Second Edition

ISBN 978-7-111-21404-5

Ⅰ. 应…　Ⅱ. ①尼…　②肖…　③丁…　Ⅲ. 数理逻辑－教材　Ⅳ. O141

中国版本图书馆 CIP 数据核字(2007)第064021号

机械工业出版社(北京市西城区百万庄大街22号　邮政编码　100037)

责任编辑：王春华

三河市明辉印装有限公司印刷・新华书店北京发行所发行

2007年7月第1版第1次印刷

184mm×260mm・18.75印张

定价：38.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：(010)68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭櫫了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业

的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

译 者 序

本书是介绍数理逻辑的基础教材。与一般的数理逻辑教材相比，本书有两个较为显著的特点。其一，本书花了很大篇幅讨论逻辑在计算机科学中的应用，例如消解定理证明、逻辑式程序设计和非经典逻辑(模态逻辑和直觉主义逻辑)，而且所用的方法与数理逻辑的经典著作有很大不同，更加适合研究计算机理论的读者，可以帮助读者更好地理解计算机理论中的许多概念，是一本真正面向计算机科学的逻辑著作。其二，本书在每章最后给出了一些进一步阅读建议，并在参考文献中对所列书目逐一做了简短有趣的点评，十分有利于读者的深入学习。

本书作者 Anil Nerode 和 Richard A. Shore 是国际知名逻辑学家、美国康奈尔大学教授，Richard A. Shore 还是上一届符号逻辑协会(the Association for Symbolic Logic)主席。

本书曾在南京大学作为研究生教材使用多年，取得了很好的效果。鉴于本书的前两版(英文版)在中国数学和计算机界取得的积极反响，我们决定将它译成中文。在本书的翻译过程中我们得到了 Nerode 教授和 Shore 教授的大力支持，在此谨表衷心的感谢。根据本书作者发来的最新勘误表，我们在译稿中对这些错误全部作了改正。此外，我们对翻译过程中发现的少量其他错误也作了修正。

全书由丁德成教授统稿。其中，第一、二章由徐亚涛翻译，第三、四、五章由吴永成翻译，第六章、前言、绪论、附录和参考文献由金陈园翻译。译者排名按照章节顺序，不分先后。

译 者

2007 年 1 月于南京大学

中文版序

本书讲述了逻辑及其在计算机科学中的应用。在当今这个信息时代，扎实的逻辑与逻辑应用功底同时惠及数学和计算机工作者。本书的前两版(英文版)在中国的数学和计算机界取得了积极反响，因此，对于本书被翻译成中文我们感到特别高兴。我们希望该中文版能够推动逻辑研究与应用的发展，同时我们也祝愿中国在逻辑、可计算性和计算机科学等领域不断取得进步。很多老师和学生使用过本书前两版并向我们提出了意见和建议，我们对此表示感谢并在此中文版里采纳了很多有益的建议。最后，我们特别感谢丁德成教授和他的学生徐亚涛、吴永成、金陈园，感谢他们花费时间和精力将我们的教材介绍给了更广大的中国读者。

Anil Nerode, Richard A. Shore
2006 年 11 月于美国纽约伊萨卡岛

附原文

This book integrates logic with its computer science applications. In this Age of Information, a firm command of logic and its applications benefits both mathematicians and computer scientists. We have been very pleased with the positive reception accorded the first two (English) editions of this book by the mathematics and computer science communities in China. It is therefore especially gratifying to have it translated into Chinese. We hope that this Chinese edition will be of service in promoting the study and applications of logic and we look forward to the continuing development of logic, computability and computer science in China. We also thank all those, teachers and students, who used the previous editions and send us their comments and suggestions. Many of these have been incorporated into this edition. Finally, we are grateful to Prof. Ding Decheng and his students Xu Yatao, Wu Yongcheng and Jin Chenyuan who have taken the time and effort to make our textbook available to a wider Chinese audience.

Anil Nerode
Richard A. Shore
Cornell University
Ithaca NY USA
November, 2006

前　言

我们写这本书的目的是提供一本合适的数理逻辑初级教程，能比传统的教科书更加适应近年来逻辑在计算机科学中应用的迅速增长。因此，我们在题材的选择上受到这些应用的重要影响。当然，本书覆盖了传统的基本内容：语法、语义、可靠性、完全性和紧致性，以及一些更高级的结果，例如斯科朗-勒文海姆(Skolem-Löwenheim)定理和厄布朗(Herbrand)定理。然而，这本书的很大一部分是讨论非传统的内容。消解定理证明在逻辑的处理中，尤其在逻辑式程序设计和 PROLOG 的应用中，起着重要的作用。因此，本书广泛地讲述了这三个主题的数学基础。此外，本书还加进了两章非经典逻辑——模态逻辑和直觉主义逻辑——它们在计算机科学中变得日益重要。对于这些逻辑，我们(通过克里普克(Kripke)框架)论述了它们语法和语义的基本内容。在这两种情形中，我们用修正的经典逻辑的表方法研究形式证明、可靠性和完全性，指出了怎样使它适用于其他多种特殊类型的模态逻辑。一些更高级的主题(包括非单调逻辑)在介绍非经典逻辑的章节以及逻辑式程序设计和 PROLOG 的内容中做了简要的介绍。

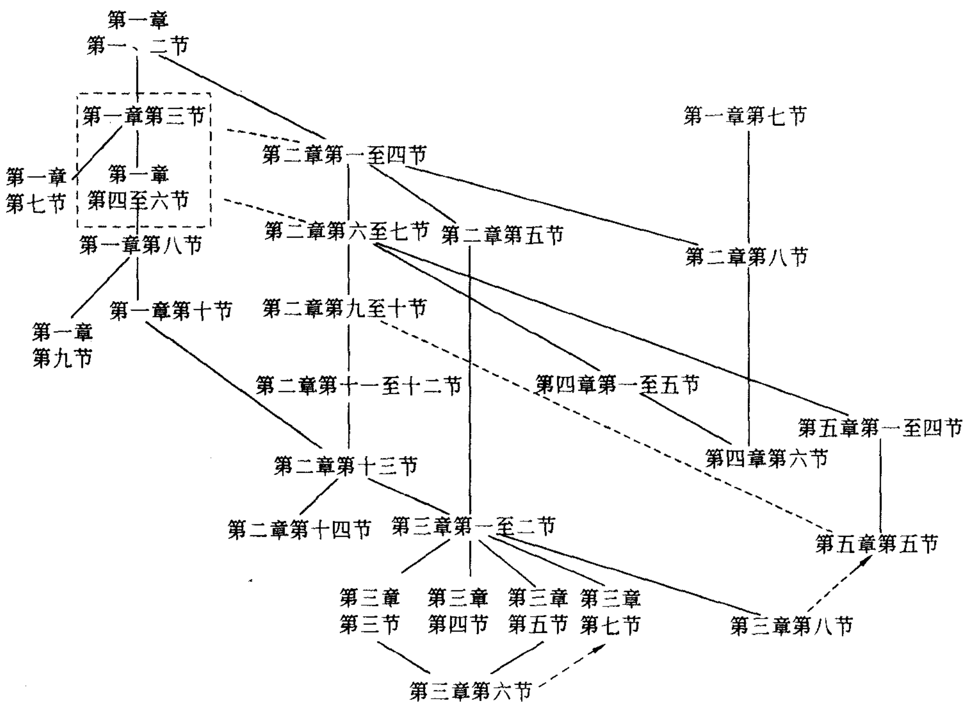
本书是为数学系和计算机科学系高年级的本科生和低年级的研究生而写。我们假定读者学过任何一门代数或理论计算机科学的入门课程，了解这些课程提供的抽象推理，并且基本熟悉这些课程用到的非形式化的数学概念和论证。本书在开篇第一章的定义 1.1 中，给出了关于序和树的基本的集合论术语。在第六章中对基础集合论进行了系统的论述。这些内容可以作为必要时的参考，也可以放在课程的开头或结尾作为课程的一部分。

如果本书作为高年级本科生的教程，第一章到第三章的所有主要内容和 PROLOG 程序设计的相当一部分可以在一学期每周三小时的课程中讲完。如果按照这种方式授课，我们建议每周用一个额外的时间处理作业中的问题和进行编程指导。关于消解定理证明和逻辑式程序设计的内容，当然也可以用有关模态逻辑和直觉主义逻辑的章节来代替，从而构成一门完全不同的课程。作为两个季度的课程，可以在第一种建议中仅仅加进一种非经典的逻辑。我们有意把这两章处理成相互独立的章节，以便于上述选择。然而，这两章的处理有很多相似之处，所以如果两章都讲的话，第二章的相应部分可以更快地讲完。作为研究生教材，整本书的主要内容可以在一个学期之内讲完。

这本教材在引入谓词逻辑之前完整地介绍了命题逻辑。然而，根据学生的背景和教师的偏好，可以重新整合这种安排。事实上，作为研究生教材或者对于了解命题逻辑与真值表的学生，这种整合可能更好。如果这么做的话，就从第一章第一、二节开始，然后转到第二章第一至四节。如果计划介绍 PROLOG 并论述逻辑式程序设计的数学基础，可以用第二章第五节来解释 PROLOG 的语法和语义。为了让学生开始进行实际的编程，在这一节(还有第一章第十节)之后，可以非形式化地介绍 PROLOG 的执行方法和程序设计。当然其理论基础要等到我们形式化地论述了消解方法和合一算法之后。不管用不用逻辑式程序设计，现在都能够(基于表方法)证明第二章第六至七节谓词逻辑的可靠性和完全性定理并在第二章第十节完成厄布朗定理的证明。我们介绍了一个希尔伯特式的证明系统，但是没有用它来证明第一章第七节和第二章第八节中的结果。

作为 PROLOG 的基础，消解式证明系统与其命题情形的论述联系更加紧密，因为基本的定理证明是根据厄布朗定理把谓词逻辑中的结论归约到相应的命题逻辑中的结论。因此，如果要涵盖这些内容，有必要先讨论命题逻辑的情形。介绍 PROLOG 要用到的章节是第一章的第八节、第十节以及第二章的第十一至十三节。第一章的第九节所考虑的加细消解是真正的备选内容。线性消解(第二章第十四节)与 PROLOG 有很大关系，而完全性定理是本书中最难的部分之一。因此，第三章第一节对于 Horn 逻辑和 PROLOG 的结果提供了一个替代的研究方法，这种方法只用到线性消解的基本定义(第二章的定义 14.1 ~ 14.3)。

下面给出了本书(除了第六章外)各章节之间的逻辑依赖关系示意图。除非用箭头额外标出，否则依赖关系都是从右向左的。虚线(例如经典逻辑中从命题逻辑到谓词逻辑的那条线)表示相关但不严格的逻辑依赖。应该指出，最前面的一节，即第一章第一节，仅仅集中了一些在后面很多地方会用到的关于序和树的定义和事实。这个内容可以放在开头讲，也可以在用到的时候加以穿插。尤其库尼西(König)引理(第一章的定理 1.4)直到第一章第四节讲完全系统表的时候才用到。类似地，第六章中的集合论的内容(第二版新增的内容)可以在任何时候讲述。只有第六节需要假定读者熟悉谓词逻辑，那里简单地列举了集合论公理的形式化版本。在这一章中，第一至五节和第七节包含了本书和大多数本科课程中所需的集合论的基本知识。其余部分，即第八至十一节，介绍了无穷的(甚至是不可数的)序数、基数和超穷递归理论，还有选择公理的一些不同的版本。对于数学系和计算机系的大部分研究生课程来说，这些集合论的背景知识应该足够了，但这些在本书的其他部分是用不到的。



在本书的很多地方，某些段落或整节对于后面的内容不是必要的，这些内容用星号(*)做了标记，以此表明它们脱离了本书内容的主线。另外，对于上面指出的内容安排有三个可能的例外，这些例外和那些不包含很多 PROLOG 内容的课程尤其相关。

第一个例外还有一点需要注意。本书对逻辑所做的基本研究不包含一个专用的等号谓词。尽管已不再是数理逻辑教材的通常做法，但对于消解定理证明、逻辑式程序设计以及PROLOG等主题，这是一个恰当的处理方法。因此，在第三章的第五节分析了等号谓词，一方面把它视为特殊的谓词，其语义解释是真正的相等，从而满足相应的逻辑公理模式，另一方面把它作为一个普通的谓词，在所考虑的每一个系统中添加适当的等词公理。然而，把第三章第五节中定义5.3之前的相关内容和其中关于等号解释的可靠性和完全性定理的证明移到第二章第七节之后也是相当可行的。

第二个例外是关于谓词逻辑永真性不可判定的丘奇(Church)定理的证明。在第三章的第八节中给出的一个证明，甚至可应用于PROLOG所表示的谓词逻辑部分。但是，在第三章第五节的习题3中我们指出，对证明稍加改动即可绕开PROLOG的概念和方法，从而可在第二章第七节之后给出丘奇定理的一个证明。

最后，在第三章第七节中定义7.6之前引入的非单调逻辑独立于逻辑式程序设计和PROLOG的内容。第三章第七节的剩余内容是用非单调逻辑分析带否定的逻辑式程序设计的稳定模型。此外，作为更加独立的内容，第三章第七节的习题8~9给出了非单调逻辑在图和偏序上的其他应用。

应该指出，在第四章和第五章关于模态逻辑和直觉主义逻辑的基本内容中有相当多的重叠。实际上，一个无重叠的论述是可能的，但那样做会降低易读性。我们分别编写这两章，以便于读者独立地阅读其中之一。对于需要深入了解这两种逻辑的读者，我们在第五章第六节比较了经典逻辑、模态逻辑和直觉主义逻辑，试图指出这些逻辑的相似和不同之处。

本书几乎每一节的最后都有很多习题，这其中也包含一些PROLOG的程序设计问题，有的是作理论分析，有的是具体执行。特别地，有一系列关于《圣经》人物的家谱数据库的习题，这个家谱是根据《旧约-历代记》编写的。我们还将它列在附录B中。(有电子版可用，请发送电子邮件至 shore@ math. cornell. edu 索取。)我们给出的这些习题可以用作使用类似数据库的范例，或者经过适当的修改后可以用于其他情形。然而这不是一本PROLOG程序设计的书，在讲授这些内容的同时，我们总是选择一本标准的PROLOG程序设计的教材作为补充，这些书目列在第三章最后的进一步阅读建议中。本书内容并不局限于特殊版本的PROLOG语言；我们仅使用标准的语法和讨论典型的执行方法。我们自己曾用过多种执行方法和平台。程序运行过程的打印件来自PC上运行的ARITY PROLOG。

当(偶尔)引用当代文献中的结果时，我们一般给出其出处。然而，由于这是一本基础的教科书，除了一些按通常做法以姓名命名的定理以外，我们并没有针对每个标准结果指明它的发现者。但是，在附录中给出了一个逻辑学的简史，这应该会让学生对于这门学科的发展有一个感性认识。另外，在每一章的最后给出了可能对学生或教师有用的进一步阅读建议。最后，在本书的末尾根据主题给出了相当多的参考文献。

多年以来，本书的部分内容以各种形式出现。关于经典逻辑和集合论的较早的版本出现在Nerode多年前在康奈尔大学的讲义中。这些内容中的一部分，由George Metakides独立地重新编写，形成了与Nerode合作的英文讲义和他自己的在佩特雷大学的希腊文讲义。Nerode [1990, 4.2]和[1991, 4.4]包含了基于表方法的直觉主义逻辑和模态逻辑处理的早期版本，这两种方法分别于1988年和1989年在Montecatini Terme和Marktoberdorf所做的演讲中提出。我们的消解方法也受到Richard Platek在康奈尔大学关于程序验证课程的影响。

过去的几年中，这些内容的较新版本已经被很多数学系和计算机科学系的教师阅读和使用过，他们提出的意见和建议让我们受益匪浅。我们感谢 Uri Abraham(以色列班固利恩大学数学系)、John Crossley(澳大利亚莫纳什大学数学和计算机科学系)、Ward Henson(乌尔班纳伊利诺斯大学数学系)、Nils Klarlund(丹麦奥尔胡斯大学计算机科学系)、Dexter Kozen(康奈尔大学计算机科学系)、George Metakides(希腊佩特雷大学，欧洲经济共同体信息技术研究所)和 Jacob Plotkin(密歇根州立大学数学系)。这一版第一章所采用的定理 4.11 及其证明由 Perry Smith(《Math. Reviews》杂志)提供。Bakhadyr Khussainov 指出了上一版中第五章的定理 2.20 的一处错误。Warren Goldfarb(哈佛大学哲学系)帮助弥补了很多在历史附录里的缺陷。将本书的某些版本用于教学的 Wiktor Marek(肯塔基大学计算机科学系)、George Odifreddi(意大利都灵大学计算机科学系)和 Robert Soare(芝加哥大学数学和计算机科学系)提出很多(而且非常有益的)建议。还应该感谢过去几年作为我们逻辑课程助教的研究生们：Jennifer Davoren、Steven Kautz、James Lipton、Sherry Marcus 和 Duminda Wijesekera，他们提供了许多勘误和建议。

我们要感谢在过去几年里得到的资助：美国国家科学基金会(NSF)的 DMS-8601048、DMS-8902797、DMS-9204308、DMS-9503503 的资助；陆军研究办公室(ARO)在康奈尔大学数学科学研究所 ACSyAM 的 DAAG29-85-C-0018 和 DAAL03-91-C-0027 的资助和 IBM 在康奈尔大学的 Ezra 计划的设备支持。我们还要感谢 Arletta Havlik 和 Graeme Bailey 帮助我们用 TEX 软件编辑前一版的原始文本，同时感谢 Geraldine Brady、Jennifer Davoren、Nathaniel Miller、Robert Milnikel、George Odifreddi 和 David Solomon 的校对工作。

最后，我们要把这本书献给我们的妻子 Sally 和 Naomi，感谢她们不断的支持。

康奈尔大学

Anil Nerode, Richard A. Shore

1996 年 6 月于纽约伊萨卡岛

目　录

出版者的话
专家指导委员会
译者序
中文版序
前言

绪论 …… 1
第一章　命题逻辑 …… 5
　第一节　序和树 …… 5
　第二节　命题、联结词和真值表 …… 8
　第三节　真值指派和赋值 …… 14
　第四节　命题演算中的表证明 …… 16
　第五节　表证明的可靠性和完全性 …… 23
　第六节　前件演绎和紧致性 …… 25
　*第七节　公理方法 …… 29
　第八节　消解 …… 31
　第九节　加细消解 …… 39
　第十节　线性消解、Horn 子句和 PROLOG …… 42
　进一步阅读建议 …… 50
第二章　谓词逻辑 …… 52
　第一节　谓词和量词 …… 52
　第二节　语言：项和公式 …… 53
　第三节　形成树、结构和列表 …… 57
　第四节　语义：含义与真值 …… 60
　第五节　PROLOG 程序解释 …… 64
　第六节　证明：完全系统表 …… 69
　第七节　表证明的可靠性和完全性 …… 76
　*第八节　公理化方法 …… 80
　第九节　前束范式和斯科朗化 …… 81
　第十节　厄布朗定理 …… 85
　第十一节　合一 …… 87
　第十二节　合一算法 …… 90
　第十三节　消解 …… 93
　第十四节　加细消解：线性消解 …… 99
　进一步阅读建议 …… 102
第三章　PROLOG …… 103
　第一节　SLD－消解 …… 103
　第二节　执行：搜索与回溯 …… 108
　第三节　执行的控制：cut …… 116
　第四节　PROLOG 程序终止的条件 …… 118
　第五节　相等 …… 123
　第六节　因失败而否定 …… 125
　第七节　否定和非单调逻辑 …… 133
　第八节　可计算性与不可判定性 …… 139
　进一步阅读建议 …… 144
第四章　模态逻辑 …… 146
　第一节　可能性与必然性；知识或信念 …… 146
　第二节　框架和力迫 …… 148
　第三节　模态表 …… 151
　第四节　可靠性和完全性 …… 156
　第五节　模态公理和特殊的可达关系 …… 163
　*第六节　公理化方法 …… 168
　进一步阅读建议 …… 170
第五章　直觉主义逻辑 …… 171
　第一节　直觉主义与构造主义 …… 171
　第二节　框架和力迫 …… 172
　第三节　直觉主义表 …… 178
　第四节　可靠性和完全性 …… 184
　第五节　可判定性和不可判定性 …… 190
　第六节　比较指南 …… 197
　进一步阅读建议 …… 201
第六章　集合论基础 …… 203
　第一节　集合论中的一些基本公理 …… 203

第二节　集合的布尔代数 …………………… 205
第三节　关系、函数和幂集公理 ………… 207
第四节　自然数、算术和无穷 …………… 211
第五节　替换、选择和基础 ……………… 218
第六节　谓词逻辑中的策梅洛-弗兰克尔集合论 ……………………………… 223
第七节　基数：有穷和可数 ……………… 224
第八节　序数 ……………………………… 228
第九节　序数算术和超穷归纳 …………… 232
第十节　超穷递归、选择和有秩全域 …… 235
第十一节　基数和基数算术 ……………… 237
进一步阅读建议 …………………………… 241
附录 A　历史回顾 ………………………… 243
附录 B　一个家谱数据库 ………………… 259
参考文献 …………………………………… 267

绪　论

在20世纪20年代，逻辑几乎是一个哲学家的世界，只有一小部分的数学家在浇灌数学这棵大树的逻辑的根。如今，逻辑学的主要分支——递归论、集合论、模型论和证明论，已经成为内容翔实的数学分支。自20世纪70年代以来，来自于计算机科学、人工智能和语言学的变革之风把新的种子吹进了逻辑的花园。这股清风掀开了一片布满丘壑的新大陆，这里的沃土滋养了新的逻辑分支。如果你近来关注计算机科学和语言学的国际会议，你会发现数理逻辑的语言已成为一种通用语言。数理逻辑的方法无所不在，理解新的逻辑并寻找可行的算法来实现这些逻辑的推理方法在许多学科中都扮演了核心的角色。含有重要逻辑成分的新兴领域包括：命令式、陈述式和函数式程序设计，程序验证，交互的、并行的、分布的、容错的以及实时的计算，知识库系统，演绎数据库，超大规模集成电路(VLSI)设计。现在，各种各样的逻辑也在从法律到医学等特殊领域的推理模型中扮演着关键的角色。

这些应用已经把逻辑研究的视野拓展到那些单纯考虑哲学和数学动机时所不会提出的问题和想法上。应用逻辑和应用数学一样，现在正应用于一个广泛的、重叠的但又有些不同的应用领域。这种情况的出现是为了满足商务、行政、科学技术中处理重要、实时和大型数据库时应用自动化推理的需要。对于大专院校的学生来说，数理逻辑及其应用像应用数学一样有用。今后，它对于很多定性学科，将像应用数学对于传统定量学科一样重要。

本书对经典的谓词逻辑作了严密而基本的介绍，它强调演绎是计算的一种形式。本书包括了可靠性、完全性和紧致性等标准的课题：书中的证明方法仅推导出永真的结果，所有永真的语句都是可证明的，每个无穷条公理的逻辑后承实际上是有穷条公理的逻辑后承。可靠性要求好像是显然的，然而当我们讨论PROLOG语言时可以看到，即使是这样简单的正确性要求，在具体执行中也常会因为要照顾效率而得不到满足。另一方面，完全性是联系着证明和永真性的重要结果。我们可以给出一个永真性的证明方法，它精确地抓住了一阶逻辑的语义。一个永真语句，即在每种解释中都为真的语句，在任何特殊的形式系统中总有一个证明，并且存在一个算法能够找到这样的证明。紧致性有着令人吃惊的应用：从关于有穷结构的结果推导出关于无穷结构的结果。这里仅举一个例子，它能从四色定理在有穷平面地图中成立导出它在任何平面地图中成立。我们也证明了永真性是不可判定的，即没有一个算法可以判定任意给定的语句是否永真。尽管对于一个给定的已经被确定是永真的语句φ，我们可以用一个特定的算法找到它的证明，但是对于一般情况，我们无法知道寻找过程是否是徒劳的。

第一章从经典命题逻辑的语法和语义开始，这是一种由诸如“与”、“或”、“如果”和“否”等联结词组合而成的复合语句的逻辑，但是不考虑量词“所有”和“存在”。我们给出传统的基于符号串形式的语法处理方式和基于树结构的语法处理方式。由于树已成为诸多计算机科学领域中的基本概念，所以对很多学生来说，后一种方法更容易理解(至少更熟悉)。两种方法都可以采用。然后我们引入由Beth(《Foundations of Mathematics》[1959, 3.2])和Smullyan(《First Order Logic》[1968, 3.2])提出的命题逻辑的语义表证明法。纵观这些年，我们发现表方法是学生们最容易学习、使用和记住的。用这种方法给出语句φ的一个证明，

就是在有限的时间内发现系统地搜索 φ 的反例是不可能的。通过该方法直接对于所要证明的公式 φ 的子公式所作的分析，我们大致可以看出其完全性的根据。这种系统的搜索是通过一个树构造算法来实现的。这个算法的目标是生成一棵由“φ 是假的”开始的有穷树，其每个分支上都有一个矛盾。这样的树表明，每一个从“φ 是假的”开始的分析都导致矛盾。我们称这是 φ 的一个表证明。通过系统地寻找表证明，可以证明可靠性、完全性和紧致性定理。

接下来论述了 J. A. Robinson[1965, 5.7]的定理证明的消解方法，这种方法在自动推理和定理证明中扮演着关键角色。再次证明其可靠性和完全性定理之后，我们将这一方法应用到 Horn 子句上，以发展逻辑式程序设计和 PROLOG 的数学基础(仍然在命题逻辑层次)。逻辑式程序设计是在某种特定环境下应用逻辑演绎进行程序设计的一种抽象方法。PROLOG 作为一种程序设计语言所体现的思想是：计算就是演绎。

第二章介绍了谓词逻辑的其他部分(函数和关系；变元和量词)，及其语法和语义的解释。还给出了谓词逻辑的表证明系统并证明了其可靠性和完全性。我们的方法自然地导致了厄布朗定理，这个定理在某种程度上把谓词逻辑归约到了命题逻辑。接下来，按照鲁宾逊(Robinson)的做法，在消解方法上添加了厄布朗首创的模式匹配算法，称为合一。于是得到了谓词逻辑的鲁宾逊演绎系统，这是在数字计算机上完全实现了推理机器化的第一个逻辑推理系统。用机器推理确实比人工推理更好。鲁宾逊的工作使得用数字计算机实现自动推理成为一个主要的研究领域。他的许多想法和所用的术语一直沿用至今。

第三章把消解方法特殊化到 Horn 子句上，Horn 子句是一类特殊的谓词逻辑公式，构成了逻辑式程序设计和 PROLOG 的定义域。逻辑式程序设计的谓词版本已经应用于专家系统、智能数据库和人工智能等很多领域。逻辑式程序设计有一个活跃的研究团队，并且已经成为一个独立的学科。除了把注意力限制在一类特殊的公式上，逻辑式程序设计和 PROLOG 在证明方法上也有诸多改变，以提高其计算效率。本章论述了 Horn 子句逻辑以及 PROLOG 的数学基础：语法、语义、可靠性和完全性。还探讨了 PROLOG 程序终止的证明。作为当前发展趋势的例子，简要说明了所谓的“通用逻辑程序”。用因失败而否定和稳定模型这两个工具说明在这个环境中否定的实现和语义。这个领域还在不断变化之中。它是还在发展之中的非单调推理这个学科的一个例子。与经典的情形不同，非单调逻辑中增加新的前件可能会推翻先前得到的结论。第三章第七节简单介绍了这个领域。然而我们不是程序员，除了举例阐述逻辑和数学思想的需要之外，不想涉及更多的关于 PROLOG 程序设计的内容。(PROLOG 程序设计方面的基础教程包含在本书最后的参考书目当中。)但是，我们要讨论关于 PROLOG 程序的理论上的可计算性和不可判定性。

对于一个理论来说，证明其不可判定性的标准做法可以概括为：说明如何把每个能行可计算的函数表示为逻辑公式，从而把计算这个函数的值归约为对于这个公式的实例的演绎。特殊函数的不可计算性，例如停机问题(给定一个程序，判断它是否会在某一个输入下停机)的不可计算性，转化成了对于给定公式可证明性的不可判定性。这样，通过证明 Horn 子句程序乃至 PROLOG 的标准执行都能计算所有的能行可计算函数，我们证明了 PROLOG 和 Horn 子句逻辑的不可判定性(更不用说所有的谓词逻辑了)。作为一个能行计算的算法的定义，我们使用寄存器机程序作为计算模型。因此，对于每个计算递归函数的寄存器机程序，我们用计算那个函数的编码版本的 PROLOG 程序来模拟。已经知道，其余的所有计算模型

都可以用寄存器机模拟，所以这样做能够得到所期望的可计算性和不可判定性的结果。

第四章和第五章转向非经典逻辑，它们在计算的理解和建模以及程序验证中变得日益重要。"非经典"的一个技术含义是：一个语句的真值不仅仅依赖于它的每个部分的真值，实际上，即使是简单句的真值也可能依赖于上下文、时间和信念等。尽管这不是数学中的传统观点，但它反映了许多现实生活中的情况和计算机科学中的许多重要问题。一个蕴涵式的真值通常含有时间要素。通常是在某个语境中考虑语句的真值。如果我们的知识或信念改变了，那么语句的真值可能也要改变。对于程序的分析依赖于计算机随时间变化的知识状态，依赖于什么可能会发生，什么一定会发生。在第三章简单地涉及了这样的一种逻辑（非单调逻辑，一种后面信息可以推翻前面结论的逻辑）。后面的两章系统地研究这样两种逻辑：模态逻辑和直觉主义逻辑。

直觉主义把数学中的构造观点融入逻辑之中。我们说有了 A 或 B 的证明仅当我们有了其中之一的证明。我们说有了"存在一个 x，具有性质 P"的证明，仅当我们实际给出了一个对象 c 和 c 具有性质 P 的证明。模态逻辑试图把握必要性和可能性的概念，以此为基础，分析那些带有时间的、动态的或基于信念的系统。我们在 Kripke 框架下描述这两种逻辑的语义。Kripke 框架是由一个经典模型的集合和这些模型上的一个偏序或其他某个关系构成；正是这些关系体现了克里普克语义的非经典方面。然后我们将经典逻辑的表方法一般化，它忠实地反映了 Kripke 框架所表示的语义。在我们的论述中，可靠性和完全性再次在其中扮演了核心角色。我们独立地介绍了这两种逻辑，但在第五章第六节对它们作了比较。

本书没有提到直觉主义的哲学教条，也没有谈论哲学家对于时间和必然性的分析。我们主要解释了 Kripke 框架和模态算子，用 Kripke 框架表达从部分信息中获得结果这样的观念，用模态算子表达模型之间的关系。这样的解释与直觉主义逻辑的一种很有前景的应用是一致的，这个应用是斯科特（Scott）所建议的，把直觉主义逻辑用作斯科特域和信息系统的语言，或者把它用来研究 Horn 子句逻辑，该逻辑更精致地应用了经典和直觉主义两种逻辑。它也可应用于模态逻辑，以进行程序分析和验证，这项工作起源于霍尔（Hoare）的动态逻辑，现已在许多方面得到继续发展。

我们在这一版中加入了新的一章（第六章）：集合论基础。这些内容可以作为标准的集合论符号和概念的参考，包括函数、关系、序和序列等。然而，这也是一个独立的公理集合论介绍。在这一章第一至六节，给出了集合论的公理，论述了足以形式化初等数论的基础集合论，例如，自然数作为满足著名的皮亚诺（Peano）后继公理的结构在同构意义下是唯一的，以及自然数上由归纳或递归所定义函数的存在性和唯一性。我们也给出了本书所需的集合论的基本概念，例如，函数、序列、序以及可数集合的基数等。在这一章的其余部分，建立了超穷归纳原则，讲述了序数和基数的基本理论（包括不可数的情况和它们的算术）以及选择公理的一些常见变种。总的来说，这些内容给计算机科学系和数学系绝大部分的研究生课程提供了足够的集合论背景。这一部分独立于本书的其余部分，但与第一章第一节以及历史附录中的一小部分内容重复。

最后，我们相信了解数理逻辑及其应用的发展史对于深入理解和欣赏这门学科是有重要影响的。因此，在附录中给出了从古希腊到 20 世纪中叶逻辑的起源和发展简史。其中有一部分只有在阅读正文（尤其前两章）后才能充分理解，但是在阅读前、阅读后或是阅读期间参考这部分都是有益的。这一部分旨在给读者提供一份旅行手册，一个该领域的向导。为了

补充这个向导，我们给出了相当多的历史参考文献和逻辑中许多主题的其他信息来源，包括一些在正文中没有涉及的内容。如果可能的话，我们建议读完本书的读者去读一些难度适中的历史材料、手册、综述以及教材。然而，一些新的主题也要求参考当代的文献。这个参考书目按照不同的主题分为若干部分（部分文献加了注释）。参考书目索引的安排是有规可循的。例如，Thomas[1939，1.1]指的是列在参考书目1.1“关于数学史的原始资料”中1939年出版的 Ivor Thomas 所著的《Selections Illustrating the History of Greek Mathematics with an English Translation》一书。我们在每一章的最后都给出了一些进一步阅读建议，其中的索引都来自这个参考书目。

第一章 命题逻辑

第一节 序 和 树

在开始本书的基本内容之前，首先介绍树的概念。树是一种常见的表示模式，是逻辑学和计算机科学中最重要的结构类型之一。我们期望大多数读者起码在直观上熟悉这种结构类型(需要集合论背景可以参见第六章)。图1给出了一个树的例子。

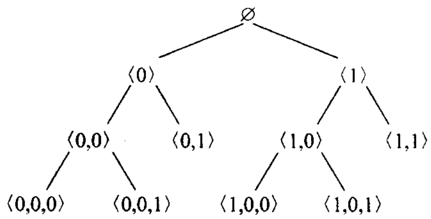


图 1

它的节点(在此例中是二元序列或者二元串)按偏序排列(序列的延伸意味着图中树的节点下降)。通常情况下，树的顶端(在其他所有节点之上)有唯一的节点(空集或空序列，∅)，叫做树的根。(自上而下地画树是为了遵循常规，而树上序的排列是为了反映序列的延伸。因此，根是序中最小的(第一个)元素，并且沿着树向下节点不断增大。)树上的节点可能有一个或者多个不可比较的直接后继。如果直接后继多于一个，则称树在该节点分叉。如果每个节点至多有 n 个直接后继，则称该树是 n-元或者 n-叉的。(图1中的树是2-元的，即通常所说的二叉树(binary tree)。)终节点，即没有后继的节点，叫做叶子。其他术语，像树的路径和层数，在直观上都有明确的意义。对于喜欢形式化定义的读者，下面给出了精确的定义。这部分内容可以暂且跳过，用到时再回过头来参考。特别是库尼西(König)引理(定理1.4)，直到第四节才会用到。

定义 1.1

(i) **偏序**(partial order)是指一个集合 S，它具有二元关系“小于”，记作 $<$，它在 S 上是**传递的**(transitive)和**反自反的**(irreflexive)：

$$x < y \text{ 且 } y < z \Rightarrow x < z \quad \text{且对任意 } x, x \text{ 不小于 } x$$

(ii) 偏序 $<$ 是**线性序**(linear order)(或简称为**序**(order))，如果它满足了**三分律**(trichotomy law)：

$$x < y \quad \text{或} \quad x = y \quad \text{或} \quad y < x$$

(iii) 线性序 S 是**良序**(well ordered)的，如果 S 的每个非空子集 A 都有一个最小元，即存在一个 $x \in A$，使得所有 $y \in A$ 都不满足 $y < x$。由该性质容易推出良序中不存在无穷递减序列，即不存在 S 中元素 x_0，x_1，…构成的集合，满足

$$\cdots < x_2 < x_1 < x_0$$

(如果存在这样一个无穷递减序列，那么由序列中所有 x_n 构成的集合 A 就没有最小元。)这个结论的逆命题见习题6。

(iv) 通常采用如下约定：

$$x \leqslant y \Leftrightarrow x < y \quad \text{或} \quad x = y$$

$$x > y \Leftrightarrow y < x$$

注意，偏序的反对称性，即 $x < y \Rightarrow y < x$ 不成立，可以由定义1.1(i)直接得到。(我们

遵循标准的数学习语，分别用“⇒”和“⇔”来表示“蕴涵”和“当且仅当”。)

备注(对那些具备一定集合论知识的读者) 除了第六章要讲的集合论基础以外，我们所考虑的每一个集合 A 都是有穷的或者可数无穷的，即 A 是有穷的或者存在一个从自然数集$\mathcal{N}$到 A 的一一映射，所以不妨假定有 A 的如下穷举：对某个 $n\in\mathcal{N}$，$\{a_i \mid i<n\}$ 或者 $\{a_i \mid i\in\mathcal{N}\}$。

定义 1.2 **树**(tree)是由偏序 $<_T$ 构成的一个集合 T(它的元素叫做**节点**(node))，其中存在唯一的最小元(叫做**根**(root))，并且每个节点的前驱构成的集合在 $<_T$ 下是良序的。

树 T 上的一条路径(path)是 T 的一个极大线性序子集。

定义 1.3

(i)归纳定义树 T 的**层数**(level)：T 的第 0 层由 T 的根组成。第 $k+1$ 层由第 k 层上所有节点的直接后继组成。(称 x 是 y 的一个后继，如果 $y\leqslant_T x$。x 是 y 的一个直接后继，如果 $y<_T x$ 并且不存在 z，使得 $y<_T z<_T x$。)

(ii)树 T 的**深度**(depth)是指树中有节点出现的最大层数 n。如果对任意自然数 n，第 n 层上都有节点，则称树 T 的深度是无穷的或者 ω。

(iii)如果每个节点至多有 n 个直接后继，则称该树是 ***n*-元**(n-ary)或者 ***n*-叉的**(n-branching)。如果每个节点都有有穷多个直接后继，则称该树是**有穷分叉的**(finitely branching)。没有后继的节点叫做**叶子**(leaf)或者**终节点**(terminal node)。

备注 我们只考虑这样的树：树上每个节点出现在第 n 层上，n 是某个自然数(如此，则该节点恰好有 n 个前驱)。

库尼西引理是关于有穷分叉树的重要结果。

定理 1.4(库尼西引理) 如果有穷分叉树 T 是无穷的，那么它有一条无穷路径。

证明 假设 T 是无穷的。归纳定义序列 x_0，x_1，…，使其构成 T 上的一条路径 P。显然，P 中第一个元素 x_0 就是 T 的根。根据假设，T 是无穷的，因此，x_0 在 T 中有无穷多个后继。假设已经定义了 P 中前 n 个元素 x_0，x_1，…，x_{n-1}，分别出现在 T 的第 0，1，…，$n-1$层上，使得每个 x_i在 T 中都有无穷多个后继。根据假设，x_{n-1} 只有有穷多个直接后继。由于 x_{n-1} 总共有无穷多个后继，所以，它的直接后继中至少有一个节点，不妨记作 y，也有无穷多个后继。现在令 $x_n=y$，则 x_n出现在 T 的第 n 层上，且它在 T 中有无穷多个后继，由此可以继续 P 的定义。 □

库尼西引理是紧致性定理在命题逻辑和拓扑学中的版本。它与命题逻辑中紧致性定理的关系可以参见第六节(尤其是定理 6.13 和第六节习题 11)，与拓扑学中紧致性定理的关系可以参见本节末尾的习题 10 和 11。

通常情况下，重要的是树形而非节点本身。为了方便讲述不同对象的相同树形排列，同时为了允许相同分支用在树中不同的地方，下面讲一讲标签树。我们把标签和树的节点联系起来。

定义 1.5 **标签树**(labeled tree)T 是指一棵带有标签函数的树 T，该函数将树中每个节点和某个对象对应起来。这个对象就叫做对应节点的**标签**(label)。

事实上，除了最初的一两次提到标签树，我们通常省去标签一词。我们所画的树已然是标签树，以使相关读者更好地理解标签树的形式定义。

附加一个线性序到整棵树上，是另外一种在树上构造更多结构的方法。考虑下面这种标

准二叉树的情形，树中每个节点都是由 0 和 1 组成的有穷序列：$S=\{0,1\}^*$ 是由这些有穷序列构成的集合。我们把一个长度为 n 的二元序列或者二元串 σ 看成一个从 $\{0,1,\cdots,n-1\}$ 到 $\{0,1\}$ 的映射。（作为专业约定，注意到集合论中对数的表示通常把 0 视为空集 $\varnothing$，把函数视为有序对的集合，例如 $\langle x,y\rangle\in f$，当且仅当 $f(x)=y$。因此，长度为 0 的序列就是处处无定义的函数。作为有序对的一个集合，该序列也等同于 $\varnothing$。有兴趣的读者可以阅读第六章中相关内容的介绍。）我们用“^”表示*毗连*（concatenation）。因此，例如，$\langle 0,1\rangle\verb|^|0$ 是 $\langle 0,1,0\rangle$，而 $\langle 0,1,0\rangle\verb|^|\langle 0,1\rangle$ 是 $\langle 0,1,0,0,1\rangle$。（注意，我们经常会混用标记，分别把 $\langle 0\rangle$，$\langle 1\rangle$ 等同于 0，1。）树上的序 $<_S$ 由函数的扩张给出，对任意的 n，$\sigma<\tau\Leftrightarrow\sigma\subset\tau$，即 $\sigma(n)=\tau(n)$，这里 σ 对 n 已经有定义。（这种情形下，我们也称串或序列 σ 是 τ 的*真初始段*（proper initial segment）。当然，σ 仅是 τ 的*初始段*（initial segment），如果 $\sigma\subset\tau$ 或 $\sigma=\tau$。）附加到树上的线性序通常是*序列上的字典序*（lexicographic ordering on sequence）：对两个序列 σ 和 τ，称 $\sigma<_L\tau$，如果 $\sigma\subset\tau$，或者序列 σ 的第 n 项 $\sigma(n)$ 小于 $\tau(n)$，其中 n 是两个序列出现差别的第一项（否则，不难得到，$\tau<_L\sigma$ 或 $\sigma=\tau$）。同样的方法可以应用到任何一棵树上，以得到树上所有节点的线性序。我们采用归纳法，在第 n 层定义线性序 $\leqslant_n$。假定 σ 和 τ 同时出现在第 $n+1$ 层上，并且分别为 σ' 和 τ' 的直接后继。如果 $\sigma'<_n\tau'$，那么 $\sigma<_{n+1}\tau$。如果 $\sigma'=\tau'$，那么就给它们的直接后继排定一个顺序，并以此确定 σ 和 τ 的关系。该序通常称为 $<_L$，也可以形象地叫做*左右序*（left to right ordering）。（这对应了两个固定长度的二元序列的排序，$\sigma<_L\tau$，如果在 σ 与 τ 序列中出现差别的第一项，σ 是 0，τ 是 1。此时称 σ 在 τ 的*左边*。）每层的左右序可以扩张成树上所有节点的一个线性序（也记作 $<_L$）：给定节点 x，y，称 $x<_L y$，如果 $x<_T y$。如果 x，y 在树序中不可比较，就寻找 T 的使得 x' 和 y'（分别为 x 和 y 的前驱）不同的最小层数。然后，按照 x'，y' 在 $<_L$ 下的排序来给 x，y 排序：$x<_L y$ 当且仅当 $x'<_L y'$。如此得到的整棵树上节点的排序，也是*所有节点的字典序*（lexicographic ordering of the nodes）。

习题

1. 给出一个有穷分叉树，使得对任意的 n，它都不是 n-叉的。
2. 给出一个深度为 3 的无穷树。
3. 证明：树中节点层数的定义是合理的，即树 T 的每个节点都恰好在某一层上。
4. 证明：树中除根外的每个节点都恰好有一个直接前驱。
5. 令 T 是一棵树。我们说 T 的两个节点 x，y 是*相邻的*（adjacent），如果其中一个节点是另外一个的直接前驱，即 $x<_T y$ 或 $y<_T x$ 并且在它们之间确实没有节点存在。证明：不存在节点序列 $x_1,\cdots,x_n(n>3)$，满足 x_i 与 x_{i+1} 相邻，$x_1=x_n$，但序列中没有其他的相等关系出现。（用图论的术语（参见第六节习题 8）来说即是，如果把图中的边定义成树中的相邻节点，那么该图是无圈的。）提示：利用习题 3。
6. 证明：如果 S 上的线性序 $<$ 中不存在无穷递减序列，那么 S 是良序的。（了解集合论的读者应当注意到，我们假设 S 是可数的。事实上，除第六章外，我们假设所有的集合都是可数的。）
7. 定义自然数集 $\mathcal{N}$ 中 n 元数组的字典序 $<_L$ 如下：$\langle x_1,\cdots,x_n\rangle<_L\langle y_1,\cdots,y_n\rangle$，如果对满足 $x_i\neq y_i$ 的最小 i，$x_i<y_i$。证明：自然数集 $\mathcal{N}$ 中数对的字典序是良序的。
8. a）证明：对任意的 n，自然数集 $\mathcal{N}$ 中 n 元数组的字典序 $<_L$ 都是良序的。

   b）证明：对任意的 $m\in\mathcal{N}$，$n<m$，所有 n 元数组构成的集合的字典序是良序的。
9. 考虑所有自然数组成的有穷序列的集合。定义序 $<$ 如下：$\sigma<\tau$ 当且仅当或者 σ 比 τ 短，或者如果 σ 不比 τ 短，则 $\sigma<_L\tau$。证明：$<$ 是良序的。

下面的两个习题是为熟悉乘积拓扑和紧致性等拓扑概念的读者设计的。

10. 证明：对于二叉树，库尼西引理等价于拓扑空间 $C=\{0,1\}^{\omega}$ 的紧致性，其中 $\{0,1\}$ 是给定的离散拓扑，而 C 是乘积拓扑。

11. 证明：对于所有的有穷分叉树，库尼西引理等价于(对有穷集合 X_i 中的每个序列)所有空间 $\prod X_i$ 的紧致性，其中 $i\in\mathcal{N}$，每个 X_i 都有离散拓扑。

第二节 命题、联结词和真值表

命题仅指语句，命题逻辑描述和研究语句相互结合形成新语句的方式，即逻辑的语法部分，它把语句当作符号串(序列)。同时我们也关心以各种方式给这些符号赋予含义，即语言的语义部分。语言的这两方面之间的关系是逻辑发展中的一个主题。对于语句内部结构的分析留到谓词逻辑那一章。现在考虑中文里从一些语句出发构造新语句的几种方式。这里所考虑的构造过程都是数学教材中出现过的基本过程。我们把从一些命题出发构造新命题的操作叫做*联结词*(connective)。在数学教材中常见的联结词有“或”、“且”、“非”、“蕴涵”和“当且仅当”。数学家最初赋予它们的含义并未明确反映它们在日常用语中的含义，经过细微改变，联结词的含义现在已经完全明确了。我们只需把它们当作数学术语的一部分。

下面介绍这些联结词的形式符号：

$\vee$“或”(析取)

$\wedge$“且”(合取)

$\neg$“非”(否定)

$\rightarrow$“蕴涵”(条件式)

$\leftrightarrow$“当且仅当”(双条件式)

在给出这些联结词的精确定义之前，先来描述怎样利用它们构造命题逻辑中的语句。任何语言的语法描述都是从其字母表开始。*命题逻辑的语言*(language of propositional logic)由下面的符号组成：

(i)*联结词*：$\vee$，$\wedge$，$\neg$，$\rightarrow$，$\leftrightarrow$

(ii)*括号*：)，(

(iii)*命题字母*：A，A_1，A_2，$\cdots$，B，B_1，B_2，$\cdots$，$\cdots$

一旦语言中包含的符号确定，我们就能描述命题语言中的语句。下面的定义是从语言中选取某些符号串，称其为命题。这是一个*归纳定义*(inductive definition)，首先描述“最短的”语句，然后描述如何根据某些确定的规则由短语句构造出更长的语句。

定义 2.1(命题)

(i)*命题字母是命题。*

(ii)*如果 α，β 是命题，则 $(\alpha\wedge\beta)$、$(\alpha\vee\beta)$、$(\neg\alpha)$、$(\alpha\rightarrow\beta)$ 和 $(\alpha\leftrightarrow\beta)$ 都是命题。*

(iii)*一个符号串是命题，当且仅当它能通过对命题字母(i)反复运用(ii)得到。*

例如，$(A\vee B)$、C、$((A\wedge B)\rightarrow C)$、$(\neg(A\wedge B)\rightarrow C)$ 都是命题，而 $A\wedge\neg$、$(A\vee B$ 和 $(\wedge\rightarrow A)$ 不是。下面的例 2.3 还会再回到这些例子。上面定义中(iii)的重要性在于它给出了将归纳法直接应用于命题的依据。下面先来介绍另外一种命题描述方法，之后再回到归纳定义的部分。

标签(二叉)树提供了一种表示命题的重要方法。不难看出，每个命题 φ 都可以表示成一棵有穷标签二叉树 T。T 的叶子由命题字母来标识。对 T 的任意非终节点，如果它由命题

α 标识，那么它的直接后继由如下命题(一个或者两个)标识，这些命题通过某个联结词联结成 α。α 的直接后继上的左右序按照分支命题的语法位置给出。继续这个过程，使得原命题 φ 成为 T 的根的标签。忽略上面的归纳定义，我们可以利用上述的标签树来定义命题，下面给出这些概念的形式化表述。

定义 2.2 形成树(formation tree)是指这样一棵二元序列的有穷树 T(以 $\varnothing$ 为根，左右序由序列通常的字典序给出)，其所有节点都用命题来标识。标签满足下列条件：

(i)叶子由命题字母标识。

(ii)如果节点 σ 由形如 $(\alpha\wedge\beta)$、$(\alpha\vee\beta)$、$(\alpha\rightarrow\beta)$ 或 $(\alpha\leftrightarrow\beta)$ 的命题标识，那么它的直接后继 $\sigma\hat{}0$ 和 $\sigma\hat{}1$ 分别由 α 和 β 来标识。

(iii)如果节点 σ 由形如 $(\neg\alpha)$ 的命题标识，那么它唯一的直接后继 $\sigma\hat{}0$ 由 α 来标识。

形成树 T 表示(represent)或者对应(associated with)一个命题，该命题标识了 T 的根。

例 2.3 如图 2 所示，通过插入适当的标签到树的节点，我们可以描述前面几个表达正确的命题所对应的形成树。

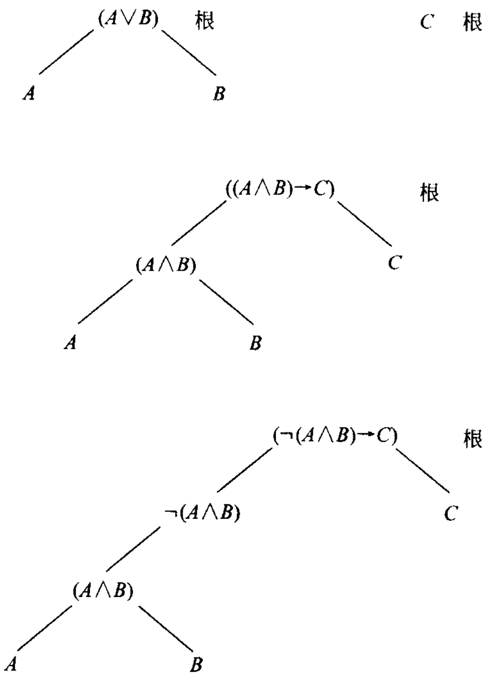


图 2

注意，这些例子中谈到的形成树都对应于某个给定的命题。在本段说明之后有一个定理指出，每个命题事实上对应着唯一的形成树。对此类定理的证明一般采用归纳法。命题的定义也是归纳定义：命题字母是基本情形，由各个联结词给出的形成规则构成了归纳步。定义中的(iii)说明所有的命题都是由此过程得到。与此类定义相对应地，也有其他利用归纳法得到的定义和证明。事实上，归纳法是我们处理大多数概念的主要方法。因此，例如要定义每个命题所对应的形成树，首先要对每个命题字母定义形成树，然后对于已有形成树定义的分支命题，指明由它们通过各个联结词得到的新命题所对应的形成树如何定义。相应地，如果要证明某个性质 P(例如，每个命题至多对应一棵形成树)对每个命题都成立，利用归纳法，步骤如下：首先证明 P 对每个命题字母都成立(基本情形)，然后证明如果 P 对命题 α，β 成立，那么它对由 α，β 通过五个基本联结词构造出的每个命题都成立。这种方法其实与自然数上的归纳证明过程并无太大区别。习题 15 给出了它在自然数上的版本。

定理 2.4 每个命题都对应唯一的一棵形成树。

证明 首先归纳证明每个命题 α 都对应一棵形成树。基本情形：α 是一个命题字母，比如说 A。在此情形下，带有标签 A 的 $\varnothing$(是根且是树上唯一节点)所构成的树即是要找的形成树。对归纳步，考虑命题 $\alpha\rightarrow\beta$。由归纳假设，α，β 分别对应形成树 T_α，T_β。对 $(\alpha\rightarrow\beta)$，把要找的形成树 $T_{(\alpha\rightarrow\beta)}$ 的根 $\varnothing$ 标识为 $(\alpha\rightarrow\beta)$，然后把 T_α，T_β 连接到根的下方，分居左右两侧。这一操作可以形式地表述如下：在 $T_{(\alpha\rightarrow\beta)}$ 中，令原 T_α 的节点 σ 为 $0\hat{}\sigma$，原 T_β 的节点 τ

为$1^\frown\tau$,而节点所对应的标签保持不变。因为T_α,T_β都是形成树,显然$T_{(\alpha\to\beta)}$也是形成树。(除了根由$(\alpha\to\beta)$标识外,其余节点的标签皆从原树继承而来。根据定义,根的直接后继分别由α,β来标识,所以用$(\alpha\to\beta)$来标识根是合理的。)对其余二元联结词的情形,可以用相同的方法处理。对$(\neg\alpha)$,我们以$(\neg\alpha)$所标识的$\varnothing$为根,只把T_α连接到根的下方,T_α中节点σ改成$0^\frown\sigma$,节点标签保持不变。注意,如果α,β对应的形成树深度分别为n和m,那么对于它们通过任一基本联结词构造出的新命题,其所对应形成树的深度为$\max\{n, m\}+1$。

下面证明每个命题至多对应一棵形成树。对命题字母,结论显然成立。作为命题字母所对应的树的根,$\varnothing$一定是由命题字母来标识,定义要求根同时是叶子,即根是整棵树。现在以$(\alpha\to\beta)$为例考虑归纳步的情形。下面这个断言至关重要:如果T是$(\alpha\to\beta)$所对应的形成树,则其根$\varnothing$的标签必是$(\alpha\to\beta)$。假设$\varnothing$还有其他标签,例如$(\gamma\vee\delta)$,那么$(\alpha\to\beta)=(\gamma\vee\delta)$。显然,在此情形下$\alpha$,$\gamma$其中一个必是另外一个的真初始段,而这与习题3和4给出的关于括号的基本事实相矛盾。对于与$(\alpha\to\beta)$具有相同字符串的命题的其他可能形式,可以类似地处理。现在,根据形成树的定义可以断言,$\varnothing$必有两个直接后继0和1,它们必由α和β分别来标识。对某个二元序列σ,T中在$i=0$或1下方的节点必定分别形如$0^\frown\sigma$或$1^\frown\sigma$。对$n=0$,1,令$T_n=\{\sigma \mid n^\frown\sigma\in T\}$有标准序,且与$T$采用相同标识。那么显然$T_0$,$T_1$分别是$\alpha$,$\beta$所对应的形成树。由归纳假设,它们是唯一的,从而$T$是唯一的,得证。

其他联结词的情形与$(\alpha\to\beta)$类似。 □

这个定理对应了命题所谓的*唯一可读性*(unique readability):在把一个命题分解到命题字母的所有方法中,只有一种方法是把命题分解成了它的各个分支。由此,只要不引起混淆,我们就习惯地省去命题记法中的括号。因此,例如,分别把$(\neg\alpha)$和$(\alpha\to\beta)$写成$\neg\alpha$和$\alpha\to\beta$。在形式上,唯一可读性提供了另外一种方法来定义命题上的函数并证明相关结论,这就是形成树上的归纳法。一般地,我们对命题所对应形成树的深度作归纳。利用形成树的优势在于,如果定义了形成树上的一个函数,那么我们就自动得到了相应命题上的函数。反过来,如果直接在命题上归纳定义一个操作,我们就模糊了这样的事实:为了保证操作的定义合理,只有一种方法能够归纳地分解给定命题。这就是我们所说的唯一可读性。读者在下一节将会看到一些在形成树上作归纳的例子。现在只需注意,定理2.4保证我们能够定义命题的深度。我们也能利用该定理来指出哪些命题字母与给定命题"相关"。

定义 2.5

(i)命题的**深度**是指其所对应形成树的深度。

(ii)命题的**支集**(support)是指用来标识其所对应形成树的叶子的那些命题字母的集合。(这个表述与命题中命题字母的语法出现相对应,可以用归纳法论证,见习题16。)

***闭包操作和归纳定义**

下面将给出另一种归纳定义,它阐明了定义2.1(iii)的作用:该条款保证了只有(i)和(ii)所生成的表达式才是命题。首先给出闭包的(代数)概念。集合S在某个(比如n-元)操作$f(s_1, \cdots, s_n)$下是**封闭的**(closed),当且仅当对每个$s_1, \cdots, s_n\in S$,$f(s_1, \cdots, s_n)\in S$。集合S在集合T中(所有)操作下的**闭包**(closure),是满足以下两条的最小集合C:

1. $S\subseteq C$。

2. 如果$f\in T$是n-元的且s_1，…，$s_n\in C$，那么$f(s_1,\ \cdots,\ s_n)\in C$。

为了说明存在这样一个最小集合，我们考虑

$$C=\bigcap\{D\mid S\subseteq D\text{且}D\text{在}T\text{中的操作下封闭}\}。$$

显然$S\subseteq C$。现在要证明C在T中的操作下封闭。若能证明之，则C显然就是我们要找的最小集合，因为它包含在每个$D\supseteq S$中，这里D是在T中的操作下封闭的集合。现在可以把命题的集合定义成(i)中命题字母的集合在(ii)中所列的$\wedge$，$\vee$，$\neg$，$\rightarrow$，$\leftrightarrow$等操作下的闭包。

现在转到语义角度，我们把命题字母的含义理解为它的真值，即它的真或假。(注意，到下一章再来分析命题的内部结构。)每个命题有唯一的*真值*(truth value)(T是真，F是假)。复合命题的真值取决于其分支命题的真值，参见图3中的*真值表*(truth table)。

定义2.6(真值表)

α	β	$(\alpha\vee\beta)$
T	T	T
T	F	T
F	T	T
F	F	F

α	β	$(\alpha\wedge\beta)$
T	T	T
T	F	F
F	T	F
F	F	F

α	β	$(\alpha\rightarrow\beta)$
T	T	T
T	F	F
F	T	T
F	F	T

α	β	$(\alpha\leftrightarrow\beta)$
T	T	T
T	F	F
F	T	F
F	F	T

α	$\neg\alpha$
T	F
F	T

图　3

正如之前所说，这些真值表所描述的联结词的含义与它们在日常英语中的含义并不完全相同。$\vee$是指可兼，或者：$\alpha\vee\beta$为真，如果α，β其中之一或者同时为真。$\rightarrow$的含义与日常用语中的“如果……则……”更是不同。在数学中，只有α为真且β为假时，$(\alpha\rightarrow\beta)$才为假，其余情形$(\alpha\rightarrow\beta)$皆为真。

下一节将形式地给出命题的真值指派，它取决于命题字母的真值。从直观上讲，给定任意一个命题，根据命题的归纳定义，我们应该清楚如何构造它的真值表，也就是把命题看作从命题字母开始一步步构造而成。例如，如图4所示的$((A\wedge B)\rightarrow C)$的真值表。

A	B	C	$(A\wedge B)$	$((A\wedge B)\rightarrow C)$
T	T	T	T	T
T	T	F	T	F
T	F	T	F	T
T	F	F	F	T
F	T	T	F	T
F	T	F	F	T
F	F	T	F	T
F	F	F	F	T

图　4.

对只有A，B，C出现的命题来说，A，B，C的八种真值组合$(2^3=8)$穷举了该命题指派的所有可能情形。$(A\wedge B)$一栏只是辅助性的，可以省去，从而得到$((A\wedge B)\rightarrow C)$的*缩略真值表*(abbreviated truth table)。上述真值表逐条罗列了命题字母A，B，C的八种真值组合，如果遵循这一罗列顺序，那么显然任意一个命题都对应了唯一的缩略真值表。

本来我们也可以从其他联结词的组合出发来定义命题。一般地，n -元联结词(n-ary connective)是这样一个函数 σ，它把 $\sigma(A_1, \cdots, A_n)$ 指派给命题 $A_1, \cdots, A_n$ 的每一个 n -元组。因此，¬是 1 -元(一元，unary)的，而∧和∨是 2 -元(二元，binary)的。n -元联结词是真值函数(truth functional)，如果 $\sigma(A_1, \cdots, A_n)$ 的真值由 $A_1, \cdots, A_n$ 的真值唯一确定。由于五个联结词的含义都是由真值表定义的，所以它们是真值函数。但是，像"因为"这样的联结词就不是真值函数。比如，令 A 代表"我早餐喝了洋李汁"，B 代表"中午地震了"，即便在 A，B 同时为真这一事件中，(B 因为 A)这一命题是否为真也是有争议的。随着 A，B 所代表内容的变化，争议可能会更加激烈。是真值函数的 n -元联结词可以通过如图 5 所示的真值表完全描述出来，真值表中的每个 $b_i(1 \leqslant i \leqslant 2^n)$要么为真，要么为假。

A_1	A_2	…	A_n	$\sigma(A_1, \cdots, A_k)$
T	T	…	T	b_1
T	T	…	F	b_2
·	·	…	·	·
·	·	…	·	·
F	F	…	F	·

图 5

反之，不同的缩略真值表(遵循约定好的 $A_1, \cdots, A_n$ 的真值排列顺序)对应不同的真值函数联结词。通过计算发现，总共有 2^{2^n} 个不同的 n -元真值函数联结词。(因此，有 $12 = 16 - 4$ 个二元联结词我们没有用到。)

定义 2.7 真值函数联结词的集合 S 是**充分的**(adequate)，如果对任意给定的真值函数联结词 σ，都能找到一个由 S 中联结词构造出来的命题，使得它与 σ 具有相同的缩略真值表。

定理 2.8(充分性) $\{\neg, \wedge, \vee\}$ 是充分的。

证明 令 $A_1, \cdots, A_k$ 为互不相同的命题字母，a_{ij} 表示 $\sigma(A_1, \cdots, A_k)$ 的真值表中第 i 行第 j 列所对应的真值(T 或 F)，如图 6 所示。假设最后一列至少出现一个 T。

A_1	…	A_j	…	A_k	…	$\sigma(A_1, \cdots, A_k)$
						b_1
						b_2
						·
						·
		a_{ij}				b_i

图 6

对任意的命题 α，令 α^T，α^F 分别为 α，$\neg\alpha$。对第 i 行，用 a_i 表示合取式(conjunction)($A_1^{a_{i1}} \wedge \cdots \wedge A_k^{a_{ik}}$)。令 $i_1, \cdots, i_m$ 是在最后一列中出现 T 的那些行。析取式(disjunction)($a_{i_1} \vee \cdots \vee a_{i_m}$)即是我们要找的命题。该命题具有如图 6 所示的真值表，证明留作习题 14。(注意，考虑到可读性，我们在命题表示中省去了很多括号，并且约定*右结合*(right associativity)，就是把 $A \wedge B \wedge C$ 看作($A \wedge (B \wedge C)$)。)此外，我们还利用集合论中的惯用术语来表述命题集合上的析取。因此，刚才构造的析取式可以写成 $\vee\{a_i \mid b_i = T\}$。 □

例 2.9 上述的证明过程可以看作仅利用$\wedge$，$\vee$ 和$\neg$这三个联结词来构造一个命题，使其具有如图 7 所示的真值表。

	A	B	C	?
1	T	T	T	T
2	T	T	F	F
3	T	F	T	F
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	F
8	F	F	F	T

图 7

首先观察最后一列真值为 T 的那些行。对其中每一行，找到一个命题使其在该行指派下为真，而在其余行指派下为假。我们最终所要找的命题就是那些相关行(在本例中指第 1，5，8 行)所对应的命题的析取式。具体到某个给定的相关行，我们对在该行真值为 T 的那些命题字母以及在该行真值为 F 的那些命题字母的否定作合取，如此即得到了仅在该行指派下为真的命题。在本例中，第 1 行对应命题$(A\wedge B\wedge C)$；第 5 行对应命题$((\neg A)\wedge B\wedge C)$；第 8 行对应命题$((\neg A)\wedge(\neg B)\wedge(\neg C))$。因此，命题$(A\wedge B\wedge C)\vee((\neg A)\wedge B\wedge C)\vee((\neg A)\wedge(\neg B)\wedge(\neg C))$具有如图 7 所示的真值表。

显然，任意给定一个命题 α，都能构造出其真值表，然后按照上述过程找到另一个与其具有相同真值表的命题，该命题是通过由命题字母及其否定组成的一些合取式析取而成。与 α 具有相同(缩略)真值表的这种形式的命题叫做 α 的析取范式(disjunctive normal form，DNF)。同样也有与 α 等价的合取范式(conjunctive normal form，CNF)，见第三节习题 3。第四节习题的最后给出了另外一种寻找与 α 等价的 DNF 和 CNF 的方法。

评注 2.10 上述过程没有告诉我们最后一列全是 F 时怎么处理。参见习题 13。

推论 2.11 $\{\neg,\vee\}$是充分的。

证明 容易验证$(A_1\wedge A_2)$与$\neg((\neg(A_1))\vee(\neg(A_2)))$具有相同的真值表。因此，任意给定命题 α，首先找到 α 的 DNF，然后通过刚才的等价替换来消除所有的$\wedge$。替换后的命题仍然具有相同的真值表。 □

本节习题中证明了$\{\neg,\wedge\}$和$\{\neg,\rightarrow\}$也是充分的。如果一个集合不是充分的，那么如何证明？(参见习题 10。)

评注 2.12 根据充分性定理(定理 2.8 和推论 2.11)，理论上只需要$\neg$，$\vee$ 和$\wedge$ 这三个联结词，甚至$\neg$和$\vee$ 这两个就足够了。这样在命题定义以及许多相关定义和证明中(例如，在第四、五、六节涉及表的部分)，归纳子句就可以大大缩短。尽管如此，我们还是保留原来的五个联结词，而在某个具体的证明中，通常详细地处理一两个联结词，其余的留作习题。

习题

1. 下面哪些表达式是命题逻辑中基于命题字母 A，B，C，D 的正式(即非缩写)命题？

(a)$((\neg(A\vee B))\wedge C)$

(b)$(A\wedge B)\vee C$

(c)$A\rightarrow(B\wedge C)$

(d)$((A\leftrightarrow B)\rightarrow(\neg A))$

(e)$((\neg A)\rightarrow B\vee C)$

(f)$(((C\vee B)\wedge A)\leftrightarrow D)$

(g)$((\vee A)\wedge(\neg B))$

(h)$(A\wedge(B\wedge C))$

2. 证明：你对习题 1(a)、(b)、(f)的判断。可以根据命题的归纳定义一步步地构造出命题(只需画出命题

对应的标签形成树)；或者用归纳法证明，存在某个性质在所有命题中都成立，但在该表达式中不成立。

3. 证明：除命题字母外的所有命题都以左括号开始，以右括号结束。此外，每个命题中左右括号的数目相等。
4. 证明：在命题的任一非空真初始段中，左括号都多于右括号。
5. 找出与下列命题等价的 DNF：

   (a) $(A\to B)\to C$

   (b) $(A\leftrightarrow B)\vee(\neg C)$
6. 证明：$\{\neg,\wedge\}$是充分的。(提示：用$\neg$和$\wedge$来代替$\vee$。)
7. 证明：$\{\neg,\to\}$是充分的。
8. Sheffer 竖(sheffer stroke)是一种二元联结词($\alpha\mid\beta$)(“非既……又”)，它的真值表如图 8 所示。证明：它是充分的。(提示：用$\mid$来代替$\neg$和$\wedge$。)
9. 证明：连带否定$\alpha\downarrow\beta$(既非α又非β)是充分的。
10. 证明：$\{\wedge,\vee\}$不是充分的。

    (提示：归纳证明，由α通过$\wedge$和$\vee$两个联结词构造出的任何命题都不与$\neg\alpha$等价。)
11. 证明：$\{\vee,\to\}$不是充分的。
12. 证明：$\{\vee,\to,\leftrightarrow\}$不是充分的。
13. 阐释如何处理定理 2.8 的证明中最后一列均为 F 的情形。
14. 证明：定理 2.8 的证明(包括习题 13 考虑的情形)中构造的表达式具有我们所要的真值表。

α	β	$\alpha\mid\beta$
T	T	F
T	F	T
F	T	T
F	F	T

图 8

15. 我们说所有的命题字母都是在第 0 层构造而成。如果命题α，β是在第n层之前构造而成，我们就说$(\neg\alpha)$、$(\alpha\vee\beta)$、$(\alpha\wedge\beta)$、$(\alpha\to\beta)$和$(\alpha\leftrightarrow\beta)$是在第$n+1$层之前构造而成。定义 2.1 的(iii)说明每个命题φ都对应某个层数n。阐释如何把基于命题定义的归纳证明转化成自然数集$\mathcal{N}$上的普通归纳。

    (提示：对命题作归纳来证明所有命题具有某个性质P，相当于对自然数n作归纳来证明所有在第n层之前构造而成的命题具有性质P。)
16. 我们说每个命题字母A出现(occur)在其自身当中，并且异于A的命题字母不出现在A中。出现在$(\neg\alpha)$中的命题字母正是那些出现在α中的命题字母。出现在$(\alpha\vee\beta)$、$(\alpha\wedge\beta)$、$(\alpha\to\beta)$和$(\alpha\leftrightarrow\beta)$中的命题字母正是那些出现在$\alpha$或$\beta$(或同时出现于$\alpha$，$\beta$)中的命题字母。这个表述清楚地揭示了命题$\alpha$中命题字母$A$的语法出现的思想。

    证明：命题α的支集正是出现在α中的那些命题字母的集合。

第三节 真值指派和赋值

我们对命题逻辑的观点是这样的：命题的含义或者内容仅指它的真值。因而，命题逻辑中语义的整个概念就只是命题的真值指派。下面我们从命题字母着手。

定义 3.1 **真值指派**(truth assignment)$\mathcal{A}$是一个函数，它给每个命题字母A指派唯一的真值$\mathcal{A}(A)\in\{T,F\}$。

现在所有命题的真值都应由命题字母的真值指派决定。该决定过程与上一节所给的联结词的真值表相一致。

定义 3.2 **真值赋值**(truth valuation)$\mathcal{V}$是一个函数，它赋给每个命题α唯一的真值$\mathcal{V}(\alpha)$，使得它在每个复合命题(即带有联结词的命题)上的真值取值与某个真值表相一致。因此，比如$\mathcal{V}((\neg\alpha))=T$当且仅当$\mathcal{V}(\alpha)=F$，而$\mathcal{V}((\alpha\vee\beta))=T$当且仅当$\mathcal{V}(\alpha)=T$或$\mathcal{V}(\beta)=T$。我们称$\mathcal{V}$**使$\alpha$为真**，如果$\mathcal{V}(\alpha)=T$。

这里的基本结果即是：命题字母的真值指派唯一地决定了所有命题上的完全真值赋值。我们通过对命题深度，即命题所对应形成树的深度，作归纳来分析这一结果。

定理 3.3 给定一个真值指派$\mathcal{A}$，存在唯一的真值赋值$\mathcal{V}$，使得$\mathcal{V}(\alpha)=\mathcal{A}(\alpha)$对所有命题字母$\alpha$成立。

证明 给定一个真值指派$\mathcal{A}$，（对形成树的深度作归纳）定义所有命题上的赋值$\mathcal{V}$。首先对所有命题字母α，令$\mathcal{V}(\alpha)=\mathcal{A}(\alpha)$，这考虑到了所有深度为 0 的形成树（命题）。假设$\mathcal{V}$在所有深度至多为n的命题上已有定义，归纳步完全由各个联结词的真值表给出。例如，假设$T_{(\alpha\to\beta)}$是$(\alpha\to\beta)$所对应的（深度为$n+1$的）形成树。（正如定理 2.4 所述，它是由T_α，T_β（两者的深度最大值恰为n）构造而来。）把$\mathcal{V}((\alpha\to\beta))$定义成$F$，当且仅当$\mathcal{V}(\alpha)=T$且$\mathcal{V}(\beta)=F$。因为$\alpha$，$\beta$的深度至多为$n$，由归纳假设知，它们的赋值在$\alpha$，$\beta$上已有定义。

至此，我们定义了一个赋值$\mathcal{V}$，它扩张了$\mathcal{A}$。下面证明由$\mathcal{A}$扩张得到的任意两个赋值$\mathcal{V}_1$，$\mathcal{V}_2$必定一致。我们通过对命题深度作归纳来证明这一点：

(i)由于$\mathcal{V}_1$，$\mathcal{V}_2$都是$\mathcal{A}$的扩张，所以对所有（深度为 0 的）命题字母α，$\mathcal{V}_1(\alpha)=\mathcal{V}_2(\alpha)$。

(ii)假设对所有深度至多为n的命题α，$\mathcal{V}_1(\alpha)=\mathcal{V}_2(\alpha)$，并假设$\alpha$，$\beta$的深度至多为$n$。于是，由归纳假设知$\mathcal{V}_1(\alpha)=\mathcal{V}_2(\alpha)$，$\mathcal{V}_1(\beta)=\mathcal{V}_2(\beta)$。又因为$\mathcal{V}_1((\alpha\wedge\beta))$，$\mathcal{V}_2((\alpha\wedge\beta))$都是由$\wedge$的真值表给出，所以$\mathcal{V}_1((\alpha\wedge\beta))=\mathcal{V}_2((\alpha\wedge\beta))$。同理可证其他联结词的情形。所以$\mathcal{V}_1$，$\mathcal{V}_2$在所有命题上的赋值都一致。 □

注意，我们对α的深度作归纳所得到的$\mathcal{V}(\alpha)$的定义只依赖于$\mathcal{A}$在α的支集（出现在α中的命题字母）上的赋值。因此，上面定理的证明实际上证明了下面这个结论：

推论 3.4 如果$\mathcal{V}_1$，$\mathcal{V}_2$在α的支集（用来构造α的那些命题字母组成的有穷集合）上赋值一致，那么$\mathcal{V}_1(\alpha)=\mathcal{V}_2(\alpha)$。

定义 3.5 在命题逻辑中，命题σ**永真**(valid)，是指对任意赋值$\mathcal{V}$，$\mathcal{V}(\sigma)=T$。这样的命题也叫做**重言式**(tautology)。

定义 3.6 两个命题α，β**逻辑等价**(logically equivalent)，是指对所有赋值$\mathcal{V}$，$\mathcal{V}(\alpha)=\mathcal{V}(\beta)$。记作$\alpha\equiv\beta$。

例 3.7

(i)$(A\vee(\neg A))$和$(((A\to B)\to A)\to A)$（排中律(law of the excluded middle)和皮尔斯律(Peirce's law)）是重言式。见习题 1。

(ii)对任意命题α和α的任意析取范式β，有$\alpha\equiv\beta$。

(iii)换一种形式表述充分性定理(2.8)：给定任何命题α，可以找到一个仅使用$\neg$，$\vee$，$\wedge$的命题β，使得$\alpha\equiv\beta$。

推论 3.4 允许我们利用定义 2.5 来检验一个给定的命题是否为重言式，该推论还给出另外一种方法，即检验命题所对应的真值表最后一列是否全部为T。我们不准备进一步研究利用真值表作证明，因为对于即将学到的谓词逻辑，真值表没有用武之地。在本节末尾，给出一些定义和符号表示，它们在今后的学习中占有重要位置，并且可以自然地推广到谓词逻辑当中。

定义 3.8 令Σ为一个（可能无穷的）命题集合。我们称σ是Σ的**后承**(consequence)（记作$\Sigma\vDash\sigma$），如果对任意的赋值$\mathcal{V}$，

$$(\mathcal{V}(\tau)=\mathrm{T}\text{ 对所有的 }\tau\in\Sigma\text{ 成立})\Rightarrow\mathcal{V}(\sigma)=\mathrm{T}。$$

注意，如果 Σ 为空集，$\Sigma\vDash\sigma$（或写成$\vDash\sigma$）当且仅当 σ 永真。该定义给出了后承的语义解释。我们在后面几节将看到几种对应不同证明过程的语法解释。我们的一个主要结果就是证明后承的语法和语义解释相互等价，这一点具体体现在（第五节）可靠性定理和完全性定理当中。

定义 3.9 我们称赋值$\mathcal{V}$是 Σ 的模型(model)，如果$\mathcal{V}(\sigma)=T$ 对所有的 $\sigma\in\Sigma$ 成立。用$\mathcal{M}(\Sigma)$来表示 Σ 所有模型的集合。

备注 在定义、定理当中，我们经常用$\Rightarrow$和$\Leftrightarrow$来代替“蕴涵”和“当且仅当”。此二者并非命题逻辑中的语言符号，而是我们讨论命题逻辑时所使用的语言（元语言）符号。

命题 3.10 令 Σ，Σ_1，Σ_2 为命题的集合，并令 $\mathrm{Cn}(\Sigma)$ 为 Σ 后承的集合，Taut 为所有重言式的集合。

(i) $\Sigma_1\subseteq\Sigma_2\Rightarrow \mathrm{Cn}(\Sigma_1)\subseteq \mathrm{Cn}(\Sigma_2)$

(ii) $\Sigma\subseteq \mathrm{Cn}(\Sigma)$

(iii) 对所有 Σ，$\mathrm{Taut}\subseteq \mathrm{Cn}(\Sigma)$

(iv) $\mathrm{Cn}(\Sigma)=\mathrm{Cn}(\mathrm{Cn}(\Sigma))$

(v) $\Sigma_1\subseteq\Sigma_2\Rightarrow\mathcal{M}(\Sigma_2)\subseteq\mathcal{M}(\Sigma_1)$

(vi) $\mathrm{Cn}(\Sigma)=\{\sigma\mid\mathcal{V}(\sigma)=T$ 对所有$\mathcal{V}\in\mathcal{M}(\Sigma)$成立$\}$

(vii) $\sigma\in\mathrm{Cn}(\{\sigma_1,\cdots,\sigma_n\})\Leftrightarrow\sigma_1\rightarrow(\sigma_2\cdots\rightarrow(\sigma_n\rightarrow\sigma)\cdots)\in\mathrm{Taut}$

我们把本命题的证明留到习题 5。

命题 3.10 的最后一个断言说明，要检验 σ 是不是有穷命题集合 Σ（集合当中的命题有时叫做“前件”）的后承可以在有穷步内完成，方法就是检验（例如，利用真值表）断言(vii)的右边命题是否是重言式。但是，如果 Σ 是无穷的怎么办？我们最好学会如何证明 σ 是后承。所考虑的第一种方法就是表方法。

习题

1. 证明：例 3.7(i)中的命题是重言式。利用推论 3.4，直接验证它们在所有赋值下为真。
2. 证明：德·摩根律(De Morgan's law)对任意命题 α_1，…，α_n 成立，即

   (a) $\neg(\alpha_1\vee\alpha_2\vee\cdots\vee\alpha_n)\equiv\neg\alpha_1\wedge\neg\alpha_2\wedge\cdots\wedge\neg\alpha_n$

   (b) $\neg(\alpha_1\wedge\alpha_2\wedge\cdots\wedge\alpha_n)\equiv\neg\alpha_1\vee\neg\alpha_2\vee\cdots\vee\neg\alpha_n$

   （提示：不必写出真值表。直接讨论析取与合取的真值条件。）
3. 命题称作文字(literal)，如果它是命题字母或其否定。命题 α 称作合取范式(CNF)，如果存在文字 $\alpha_{1,1}$，…，α_{1,n_1}，$\alpha_{2,1}$，…，α_{2,n_2}，…，$\alpha_{k,1}$，…，α_{k,n_k}，使得 α 为

   $(\alpha_{1,1}\vee\alpha_{1,2}\vee\cdots\vee\alpha_{1,n_1})\wedge(\alpha_{2,1}\vee\alpha_{2,2}\vee\cdots\vee\alpha_{2,n_2})\wedge\cdots\wedge(\alpha_{k,1}\vee\cdots\vee\alpha_{k,n_k})$。

   证明：每个命题都有一个等价的合取范式（即有相同的真值表）。（提示：考虑$\neg\alpha$ 的析取范式，然后利用习题 2。）
4. 为下列命题寻找合取范式：

   (a) $(A\wedge B\wedge C)\rightarrow D$

   (b) $(A\wedge B)\rightarrow(C\vee D)$
5. 从适当的定义出发，给出命题 3.10 中(i)～(vii)的（简要）证明。

第四节 命题演算中的表证明

本节介绍一种可以构造命题证明的系统，此类证明都是标签二叉树，叫做表。树上的标签是标号命题(signed proposition)，即开头标有 T 或 F（我们可以认为是给命题指定了一个假

定真值)的命题。我们把节点的标签叫做表值(entries of the tableau)。形式上，我们归纳地定义(或描述如何构造)表，首先指定某些(标签二叉)树作为表(所谓的原子表)，然后根据扩展规则把简单命题的表扩展到复合命题。

构造表的思路就是从某个表值，即某个标号命题比如 $F(\neg(A\wedge(B\vee C)))$ 开始，把它分解到其各分支。我们称表值是正确的，如果关于给定命题的真值假设正确。就当前的例子而言，$F(\neg(A\wedge(B\vee C)))$ 意味着 $\neg(A\wedge(B\vee C))$ 为假。分解的指导原则如下：如果表值正确，那么由它分解出来的分支表值至少有一个是正确的。在这个例子中，先把 $F(\neg(A\wedge(B\vee C)))$ 分解到 $T(A\wedge(B\vee C))$。(若 $\neg(A\wedge(B\vee C))$ 为假，则 $(A\wedge(B\vee C))$ 为真。)然后把 $T(A\wedge(B\vee C))$ 分解到 TA 和 $T(B\vee C)$。(若 $(A\wedge(B\vee C))$ 为真，则 A 和 $(B\vee C)$ 都为真。)接下来再把 $T(B\vee C)$ 分解到 TB 或 TC。(若 $(B\vee C)$ 为真，则 B 或 C 之一为真。)

上述过程作为构造命题证明的一种方法，其主要思路如下：从某个标号命题比如 $F\alpha$ 开始，以其为树的根，然后利用上述过程把它分解到其各分支，找到导出矛盾的那一支分解(对应树上一条路径)。于是得出结论：反驳了 α 为假的原假设，从而得到 α 的证明。举个例子，假设从 $F(\neg(A\wedge\neg A))$ 开始，按照上述过程进行分解(用 $\neg A$ 代替 $(B\vee C)$)。得到 TA 和 $T\neg A$，然后把 $T\neg A$ 分解到 FA。现在得到两个表值，分别是 A 为真和为假。这正是我们要找的矛盾，于是证明了 $\neg(A\wedge\neg A)$ 是永真命题。

表的归纳定义的基本情形是由一些(标签二叉)树组成，如图 9 所示。它们叫做*原子表*(atomic tableau)，对任意命题 α，β 以及任意命题字母 A 成立。下面给出表的形式定义。

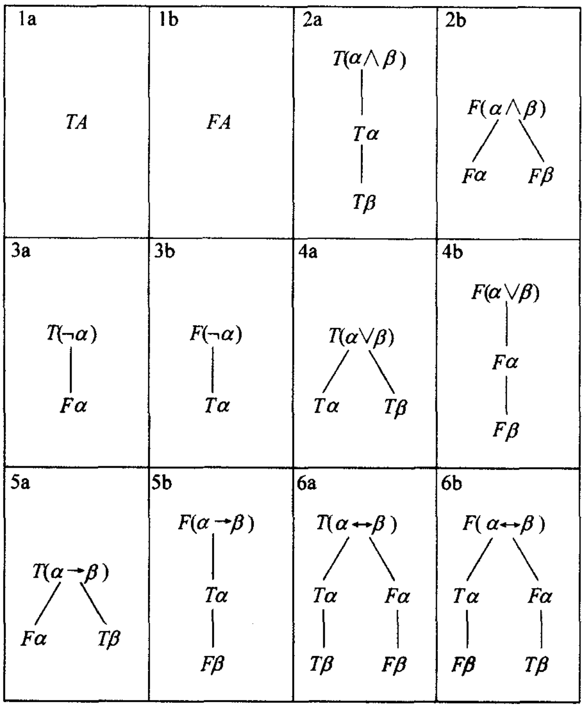


图 9

定义 4.1(表)　**有穷表**(finite tableau)是一棵二叉树，由(称作表值的)标号命题所标识，并且满足下面的归纳定义：

(i)所有原子表都是有穷表。

(ii)令 τ 是一个有穷表，P 是 τ 中一条路径，E 是出现在 P 上的 τ 的一个表值。如果 τ' 是由 τ 通过把唯一的以 E 为根的原子表添加到路径 P 的末端得到的，那么 τ' 也是有穷表。

如果 τ_0，τ_1，…，τ_n，…是有穷表的一个(有穷或者无穷)序列，使得对每个 $n\geqslant 0$，τ_{n+1} 是由 τ_n 应用(ii)构造而来，那么 $\tau=\bigcup\tau_n$ 是一个**表**。

这个定义描述了所有可能的表。本节只需要有穷表(见定理 4.11)，但是在谓词逻辑中，甚至在下一节考虑带有(无穷多)前件的命题逻辑时，无穷表都是必须要用到的。

每个表都是分解某个命题的一种方法。其主要思路如下：如果可以假设通往表值 E 的路径上的所有表值都是正确的，那么在沿 E 继续通往下一层的路径中，也有一条是正确的。

为了说明这一思路是可以实现的，首先考虑原子表，例如，考虑(5a)。如果 $\alpha\to\beta$ 为真，那么它的其中一个分支也为真：α 为假或者 β 为真。同样地考虑(4a)，如果 $\alpha\vee\beta$ 为真，那么 α，β 其中之一为真。其余的原子表可以用同样的方法进行分解。下一节将给出这个思路的形式化表述，即表的可靠性定理。关于表的另外一个主要定理就是完全性定理，其主要思想是：如果 α 永真，那么对给定的标号命题 $F\alpha$，所有可能的分解都导致矛盾。这就构成了 α 的一个证明。为了实现这一思想，我们必须对给定的根，给出一个系统的方法来生成表，使之包含所有可能的分解过程。在此之前，先来看几个例子。

例 4.2　我们期望从 $F(((\alpha\to\beta)\vee(\gamma\vee\delta))\wedge(\alpha\vee\beta))$ 开始构造一个表。仅有一个原子表以此表值为根，即如图 10 所示的原子表。

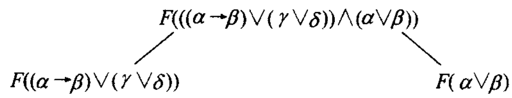


图　10

现在该表除根以外有两个表值，我们可以任选一个，根据归纳定义来构造更大的表。(我们本可以再次使用根表值，但那样做意义不大。)两种可能情形分别如图 11a、图 11b 所示。

我们也可以同时对两种可能进行分解，从而得到如图 11c 所示的表。

在图 11c 中，我们可以(忽略重复)从 $F(\alpha\to\beta)$ 或 $F(\gamma\vee\delta)$ 中任选一个作为表值继续分解。包含以上两个表值之一的路径只有一条，即末端为 $F(\gamma\vee\delta)$ 的那条。因此，无论继续分解哪个表值，相应的原子表都要添加到该路径上。如果选择 $F(\alpha\to\beta)$，所得表如图 12 所示。

注意，每当我们选择一个表值进行分解，即将其对应的原子表添加到所选路径的末端，作为原子表的一部分，该表值就在所选路径上重复出现一次。为了便于标记，在画表时我们通常省去第二次出现，但是在形式定义中保留重复出现。(因为在谓词演算中重复出现是必要的，所以我们在形式定义中保留了它们。)

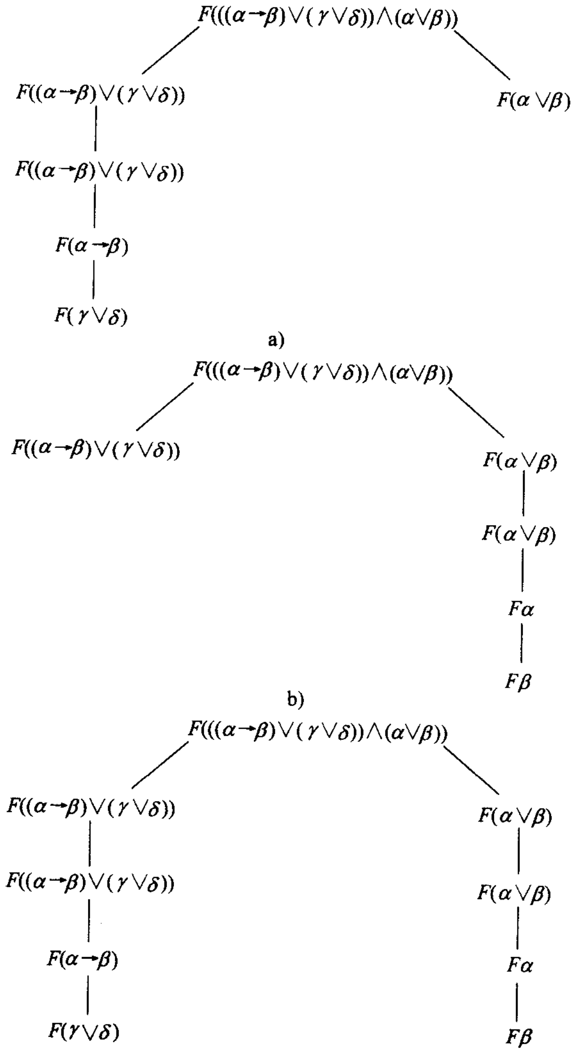

$F(((\alpha\to\beta)\vee(\gamma\vee\delta))\wedge(\alpha\vee\beta))$
$F((\alpha\to\beta)\vee(\gamma\vee\delta))$
$F(\alpha\vee\beta)$
$F((\alpha\to\beta)\vee(\gamma\vee\delta))$
$F(\alpha\to\beta)$
$F(\gamma\vee\delta)$
a)
$F(((\alpha\to\beta)\vee(\gamma\vee\delta))\wedge(\alpha\vee\beta))$
$F((\alpha\to\beta)\vee(\gamma\vee\delta))$
$F(\alpha\vee\beta)$
$F(\alpha\vee\beta)$
$F\alpha$
$F\beta$
b)
$F(((\alpha\to\beta)\vee(\gamma\vee\delta))\wedge(\alpha\vee\beta))$
$F((\alpha\to\beta)\vee(\gamma\vee\delta))$
$F(\alpha\vee\beta)$
$F((\alpha\to\beta)\vee(\gamma\vee\delta))$
$F(\alpha\vee\beta)$
$F(\alpha\to\beta)$
$F\alpha$
$F(\gamma\vee\delta)$
$F\beta$
c)

图 11

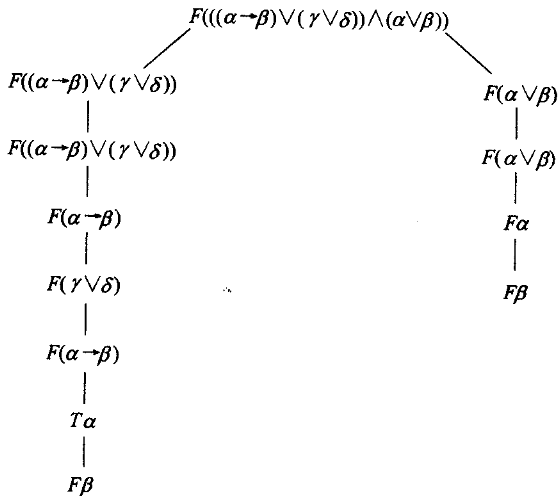

$F(((\alpha\to\beta)\vee(\gamma\vee\delta))\wedge(\alpha\vee\beta))$
$F((\alpha\to\beta)\vee(\gamma\vee\delta))$
$F(\alpha\vee\beta)$
$F((\alpha\to\beta)\vee(\gamma\vee\delta))$
$F(\alpha\vee\beta)$
$F(\alpha\to\beta)$
$F\alpha$
$F(\gamma\vee\delta)$
$F\beta$
$F(\alpha\to\beta)$
$T\alpha$
$F\beta$

图 12

现在我们希望描述构成证明的那些表，以及从给定的标号命题出发生成这些表的系统过程。我们要用到一些辅助概念：

定义 4.3 令 τ 是一个表，P 是 τ 上一条路径，E 是出现在 P 上的一个表值。

(i) E 在 P 上**已约化**(reduced)，如果以 E 为根的原子表中某条路径上的所有表值都出现在 P 上。(例如，TA 和 FA 对所有命题字母 A 都已约化。$T\neg\alpha$ 和 $F\neg\alpha$(在 P 上)已约化，如果 $F\alpha$ 和 $T\alpha$ 分别出现在 P 上。$T(\alpha\vee\beta)$ 已约化，如果 $T\alpha$，$T\beta$ 之一出现在 P 上。$F(\alpha\vee\beta)$ 已约化，如果 $F\alpha$ 和 $F\beta$ 同时出现在 P 上。)

(ii) P 是**矛盾的**(contradictory)，如果对某个命题 α，$T\alpha$ 和 $F\alpha$ 都是 P 上的表值。P 是**完成的**(finished)，如果它是矛盾的或者 P 上所有表值都已约化。

(iii) τ 是**完成的**，如果 τ 中所有路径都是完成的。

(iv) τ 是**矛盾的**，如果 τ 中所有路径都是矛盾的。(此时当然也是完成的。)

例 4.4 图 13 给出了一个带有三条路径的完成表。最左边的路径是矛盾的，其余两条不是。

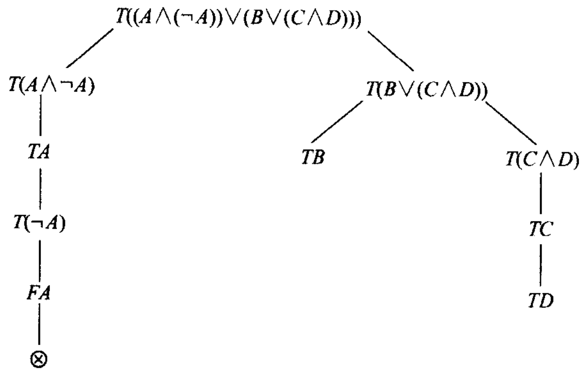


图 13

现在我们可以用如下方式定义 α 的表证明：如果假设 α 为假，那么总能导出矛盾。

定义 4.5 命题 α 的**表证明**(tableau proof)是指以 $F\alpha$ 为根表值的矛盾表。一个命题是**表可证**(tableau provable)的，记作 $\vdash\alpha$，如果它有一个表证明。

命题 α 的**表反驳**(tableau refutation)是指由 $T\alpha$ 开始的矛盾表。一个命题是**表可反驳**(tableau refutable)的，如果它有一个表反驳。

例 4.6 皮尔斯(Peirce)律。图 14 给出了皮尔斯律的一个实例的表证明。注意，我们省去了要约化表值的重复出现。我们在路径的末端画⊗来表示该路径矛盾。

对于下述所有关于表证明或逻辑真值(或同时有关两者)的定义和定理，大多分别存在其关于表反驳和逻辑假值的对偶定义或定理。我们把这些对偶命题的表述留给读者。

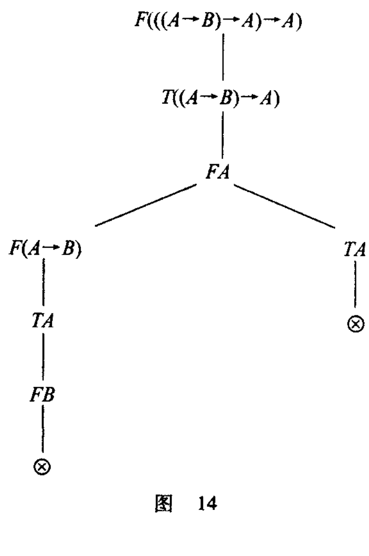


图 14

下一步就是观察对任给的标号命题，是否存在以其为根表值的完成表。下面将描述一种简单的生成此类表的系统过程。定理 4.11 表明由此过程生成的表总是有穷的。

定义 4.7(完全系统表) 令 R 是一个标号命题。我们归纳定义以 R 为根表值的**完全系统表**(complete systematic tableau, CST)。首先令 τ_0 为以 R 为根的唯一原子表。假设 τ_m 已经定义。令 n 是 τ_m 中满足下述条件的最小层数：该层存在一个表值在 τ_m 的某条非矛盾路径上未约化，并令 E 是第 n 层上最左边的未约化表值。现在令 τ_{m+1} 是通过以下过程得到的表：把以 E 为根的原子表添加到 E 在其上未约化的 τ_m 的所有非矛盾路径的末端。如此构造出的 τ_m 序列的并集就是我们要找的完全系统表。

定理 4.8 所有 CST 都是完成的。

证明 考虑出现在完全系统表 τ 的第 n 层，并且出现在 τ 的某条非矛盾路径 P 上的任意表值 E。τ 中至多有有穷多个表值在第 n 层或第 n 层以上。因此，τ 中第 n 层或第 n 层以上的所有表值都会在构造过程中的某一点出现。即存在 m_0，使得对所有 $m \geqslant m_0$，τ_m 的前 n 层和 τ 的前 n 层完全相同。现在，对 $m \geqslant m_0$，限制到 τ_m 上的 P 是 τ_m 中包含 E 的一条路径。在构造 CST 的第 $m(m \geqslant m_0)$ 步，如果 E 在 P 上尚未约化，那么我们就在第 $k(k \leqslant n)$ 层上选取一个未约化表值进行约化。继续该构造过程，至多有穷步之后我们就可以约化 E 了。 □

从直观上讲，证明过程应该都是有穷的，这似乎与无穷表的存在相冲突。然而根据库尼西引理，我们可以只关注有穷矛盾表。

定理 4.9 如果 $\tau = \cup \tau_n$ 是一个矛盾表，那么对某个 m，τ_m 是一个有穷矛盾表。因此，特别地，如果一个 CST 是证明，那么它是一个有穷表。

证明 τ 是一个有穷分叉树。考虑 τ 中所有在其之前没有矛盾的节点所构成的子集，如果该集合是无穷的，那么根据库尼西引理，它有一条无穷路径。这与假设"τ 中所有路径都是矛盾的"相悖，所以在其之前没有矛盾的节点只有有穷多个。它们必定在 τ 的某层(比如第 n 层)之前全部出现。因此，对于第 $n+1$ 层上的每个节点，在它之前都有矛盾。此外，由于 τ 在第 $n+1$ 层的节点是有穷的，所以存在一个 m，使得 τ_m 在第 $n+1$ 层和 τ 完全一样。现在 τ_m 中每条路径 P，要么是 τ 中的路径(以层数 $\leqslant n$ 的叶子为末端)，要么是包含第 $n+1$ 层上某个节点的路径。对第一种情形，根据假设 τ 是矛盾的，所以 P 是矛盾的。第二种情形，根据所选的 n 和 m 知，P 是矛盾的。因此，τ_m 即是要找的矛盾表。

注意，如果 $\tau = \cup \tau_n$，如 CST 所定义，并且 m 是使得 τ_m 矛盾的最小值，那么我们就无法在 τ 的构造过程中扩展 τ_m。在此情形下 $\tau = \tau_m$。 □

下一节将考虑这个结论在紧致性定理的语义和语法版本中的重要性。

在本节最后，给出一个证明：每个 CST 都是有穷的。该证明涉及对命题复杂性的测度作归纳，我们把这种测度称作命题的度(degree of a proposition)。

定义 4.10 我们归纳定义**命题 α 的度**(degree of a proposition α)$d(\alpha)$。

(i)如果 α 是命题字母，那么 $d(\alpha)=0$。

(ii)如果 α 是 $\neg\beta$，那么 $d(\alpha)=d(\beta)+1$。

(iii)如果 α 是 $\beta \vee \gamma$，$\beta \wedge \gamma$，$\beta \rightarrow \gamma$ 或者 $\beta \leftrightarrow \gamma$，那么 $d(\alpha)=d(\beta)+d(\gamma)+1$。

标号命题 $T\alpha$ 或 $F\alpha$ 的度就是 α 的度。如果 P 是表 τ 中的一条路径，那么 P 的度 $d(P)$ 就是所有在 P 上未约化的 P 上标号命题的度的总和。度的概念对应了命题中联结词出现的次数。

定理4.11 所有 CST 都是有穷的。

证明 令 $\tau = \cup \tau_m$ 是定义 4.7 所述的任意 CST。我们证明 τ 中所有路径都是有穷的(事实上其长度至多为 τ 的根的度),再根据库尼西引理推出 τ 自身是有穷的。考虑 τ 中任意路径 P,它是 τ_m 中路径 P_m 的并。我们要特别注意从 P_m 到 P_{m+1} 的变化,根据定义 4.7,对 τ_m 中某个未约化的表值 E,我们把以 E 为根的原子表添加到 P_m 的末端。我们断言:$d(P_{m+1}) < d(P_m)$。由此立即推出,我们至多只能有穷次地(事实上至多 $d(\alpha)$ 次,α 是 τ 的根)把原子表添加到 P_m 的末端。因此 P 是有穷的。为了验证断言,首先注意到,把 σ 添加到路径的末端就约化了 P 上的表值 E。如此就要从路径的度中减去 $d(E)$,因为计算 P_{m+1} 的度时,E 不再是未约化表值,我们只计算除 E 之外的未约化标号命题的度的总和。因此只需对所有原子表 σ,验证其每条路径上(除根之外的)标号公式的度的总和小于 σ 根的度。这一点可由度的定义及图 9 中的原子表立即得到。 □

习题

给出下列 1~9 中命题的表证明。

1. $\wedge$, $\vee$ 的幂等性(Idempotence)和交换性(Commutativity)

   (a) $(\alpha \vee \alpha) \leftrightarrow \alpha$

   (b) $(\alpha \wedge \alpha) \leftrightarrow \alpha$

   (c) $(\alpha \wedge \beta) \leftrightarrow (\beta \wedge \alpha)$

   (d) $(\alpha \vee \beta) \leftrightarrow (\beta \vee \alpha)$

2. $\wedge$, $\vee$ 的结合性(Associativity)和分配性(Distributivity)

   (a) $((\alpha \wedge \beta) \wedge \gamma) \leftrightarrow (\alpha \wedge (\beta \wedge \gamma))$

   (b) $((\alpha \vee \beta) \vee \gamma) \leftrightarrow (\alpha \vee (\beta \vee \gamma))$

   (c) $(\alpha \vee (\beta \wedge \gamma)) \leftrightarrow ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

   (d) $(\alpha \wedge (\beta \vee \gamma)) \leftrightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$

3. 纯蕴涵律(Pure Implication Law)

   (a) $\alpha \to \alpha$

   (b) $\alpha \to (\beta \to \alpha)$

   (c) $(\alpha \to \beta) \to ((\beta \to \gamma) \to (\alpha \to \gamma))$

   (d) $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$

4. $\wedge$ 的引入(Introduction)和消去(Elimination)

   (a) $(\alpha \to (\beta \to \gamma)) \to ((\alpha \wedge \beta) \to \gamma)$

   (b) $((\alpha \wedge \beta) \to \gamma) \to (\alpha \to (\beta \to \gamma))$

5. 德·摩根律(De Morgan's Law)

   (a) $\neg(\alpha \vee \beta) \leftrightarrow (\neg\alpha \wedge \neg\beta)$

   (b) $\neg(\alpha \wedge \beta) \leftrightarrow (\neg\alpha \vee \neg\beta)$

6. 换质位(Contrapositive) $(\alpha \to \beta) \leftrightarrow (\neg\beta \to \neg\alpha)$

7. 双重否定(Double Negation) $\alpha \leftrightarrow \neg\neg\alpha$

8. 矛盾(Contradiction) $\neg(\alpha \wedge \neg\alpha)$

9. (a) $(\neg\alpha \vee \beta) \to (\alpha \to \beta)$

   (b) $(\alpha \to \beta) \to (\neg\alpha \vee \beta)$

合取范式与析取范式

回顾第三节的习题 3,命题 α 的合取范式(CNF)是指把 α 写成一些文字(命题字母或其否定)的若干组析取式的合取形式,使之与 α 等价。类似地,命题 α 的析取范式(DNF)是指把 α 写成一些文字(命题字母或其否定)的若干组合取式的析取形式,使之与 α 等价。对任意命题 α,我们都能通过如下过程找到其合取

范式与析取范式：

(i)在 α 中消去 $\leftrightarrow$：把 α 中所有从 β、γ 出发构造 $\beta\leftrightarrow\gamma$ 的地方都换成 $(\beta\to\gamma)\wedge(\gamma\to\beta)$。这样得到的与 α 等价的新命题 α_1 中不出现 $\leftrightarrow$。

(ii)在 α_1 中消去 $\to$：把 α_1 中所有从 β、γ 出发构造 $\beta\to\gamma$ 的地方都换成 $\neg\beta\vee\gamma$。这样得到的与 α 等价的新命题 α_2 中只包含 $\neg$，$\vee$ 和 $\wedge$ 三个联结词。

(iii)把 $\neg$ 分配到各个命题字母之前：把 α_2 中所有出现 $\neg\neg\beta$，$\neg(\beta\vee\gamma)$ 和 $\neg(\beta\wedge\gamma)$ 的地方分别换成 β，$\neg\beta\wedge\neg\gamma$ 和 $\neg\beta\vee\neg\gamma$。如此得到的新命题 α_3 等价于 α。

(iv)现在我们利用前面提到的结合律和分配律来得到 α_3 的等价命题，要么是一些文字的若干组析取式的合取(CNF)，要么是一些文字的若干组合取式的析取(DNF)。

下面我们给出上述过程的一个实例，即寻找命题 $\alpha=(A\to B)\leftrightarrow\neg C$ 的两种范式：

$$(A\to B)\leftrightarrow\neg C \qquad \text{(i)}$$
$$((A\to B)\to\neg C)\wedge(\neg C\to(A\to B)) \qquad \text{(ii)}$$
$$(\neg(\neg A\vee B)\vee\neg C)\wedge(\neg\neg C\vee(\neg A\vee B)) \qquad \text{(iii)}$$
$$((\neg\neg A\wedge\neg B)\vee\neg C)\wedge(\neg\neg C\vee(\neg A\vee B)) \qquad \text{(iii)}$$
$$((A\wedge\neg B)\vee\neg C)\wedge(C\vee(\neg A\vee B)) \qquad \text{(iii)}$$

现在我们可以应用第(iv)步来得到 α 的 CNF：

$(A\vee\neg C)\wedge(\neg B\vee\neg C)\wedge(C\vee\neg A\vee B)$。

我们也可以通过分配律得到 α 的 DNF：

$(((A\wedge\neg B)\vee\neg C)\wedge C)\vee(((A\wedge\neg B)\vee\neg C)\wedge(\neg A\vee B))$

$(A\wedge\neg B\wedge C)\vee(\neg C\wedge C)\vee(A\wedge\neg B\wedge\neg A)\vee(\neg C\wedge\neg A)\vee(A\wedge\neg B\wedge B)\vee(\neg C\wedge B)$。

最后一行是 α 的一个 DNF。不过它还可以简化，这要用到前面已经证明的其他规则以及缩略真值表。特别地，像 $C\wedge\neg C$ 这样的矛盾式可以从析取中消去，而像 $C\vee\neg C$ 这样的重言式则可从合取中消去。综上，α 的 DNF 可以简化如下：

$$(A\wedge\neg B\wedge C)\vee(\neg C\wedge\neg A)\vee(\neg C\wedge B)。$$

10. 利用上述过程寻找与下列命题等价的 CNF 和 DNF：

(a) $(A\to B)\leftrightarrow(A\to C)$

(b) $(A\leftrightarrow B)\to(C\vee D)$

11. 利用上面习题给出的定律证明：上述过程的每一步所生成的命题都等价于原命题 α。

第五节　表证明的可靠性和完全性

我们准备证明有效性($\vDash$)的语义概念和可证性($\vdash$)的语法概念相互等价。为此，需要分别证明：所有表可证的命题都是永真的(证明的可靠性)，所有永真命题都是表可证的(证明的完全性)。

定理 5.1(可靠性)　*如果 α 是表可证的，那么 α 是永真的，即 $\vdash\alpha\Rightarrow\vDash\alpha$。*

证明　采用反证法。假设 α 非永真。根据定义，存在一个赋值 $\mathcal{V}$，指派 F 给 α。在以下两种情况下，我们称赋值 $\mathcal{V}$ 与标号命题 E 相一致：如果 E 是 $T\alpha$ 且 $\mathcal{V}(\alpha)=T$，或者如果 E 是 $F\alpha$ 且 $\mathcal{V}(\alpha)=F$。我们证明(引理 5.2)，对任意的赋值 $\mathcal{V}$，如果它与某个表的根相一致，那么它必与该表中某条路径 P 上的所有表值相一致。对矛盾表中的任何一条路径，都不存在赋值与其相一致，因而不存在 α 的表证明。 □

引理 5.2　*如果 τ 是如定义 4.1 所给的表 $\cup\tau_n$，$\mathcal{V}$ 是与 τ 的根表值相一致的赋值，那么 τ 中存在一条路径 P，其上所有表值都与 $\mathcal{V}$ 相一致。*

证明　归纳证明：存在一个序列 $\langle P_n\rangle$，使得对任意的 n，P_n 包含在 P_{n+1} 中，且 P_n 是 τ_n 中所有表值与 $\mathcal{V}$ 相一致的一条路径。P_n 的并集就是我们要找的 τ 中的路径 P。根据假设 $\mathcal{V}$ 与 τ

的根相一致，易知归纳法的基本情形成立。考虑(6a)根表值为 $T(\alpha\leftrightarrow\beta)$ 的例子：如果 $\mathcal{V}(\alpha\leftrightarrow\beta)=T$，那么根据 $\leftrightarrow$ 的真值表定义，$\mathcal{V}(\alpha)=T$ 且 $\mathcal{V}(\beta)=T$，或者 $\mathcal{V}(\alpha)=F$ 且 $\mathcal{V}(\beta)=F$。我们把其余原子表的验证留作习题 1。

对归纳步，假设已经构造了 τ_n 中的路径 P_n，它的所有表值都与 $\mathcal{V}$ 相一致。如果 τ_{n+1} 是由 τ_n 扩展而来，但未扩展 P_n，那么令 $P_{n+1}=P_n$。如果 τ_{n+1} 扩展了 P_n，那么即是对出现在 P_n 中的某个表值 E，将以其为根的原子表添加到 P_n 的末端。由归纳假设知 $\mathcal{V}$ 与 E 相一致，类似于对基本情形的分析，得到 $\mathcal{V}$ 必与 P_n 的某条扩展路径相一致，记之为 P_{n+1}。 □

定理 5.3(完全性) *如果 α 永真，那么 α 是表可证的，即 $\vDash\alpha\Rightarrow\vdash\alpha$。事实上，任何以 $F\alpha$ 为根表值的完成表都是 α 的一个证明。特别地，以 $F\alpha$ 为根的完全系统表就是这样一个证明。*

完全系统表证明用到的主要思想包含在引理 5.4 中：对任意完成表中的任意一条非矛盾路径，我们总可以定义一个赋值使其与该路径上的所有表值相一致。

定理 5.4 *令 P 是完成表 τ 中的一条非矛盾路径。定义所有命题字母 A 上的一个真值指派 $\mathcal{A}$ 如下：*

$\mathcal{A}(A)=T$，如果 TA 是 P 上的一个表值。

$\mathcal{A}(A)=F$，其他。

如果 $\mathcal{V}$ 是扩展真值指派 $\mathcal{A}$ 得到的唯一赋值(定理 3.3)，那么 $\mathcal{V}$ 与 P 上所有赋值相一致。

证明 对 P 上命题的深度作归纳。

(i)如果 α 是命题字母且 $T\alpha$ 出现在 P 上，那么根据定义有 $\mathcal{V}(\alpha)=T$。如果 $F\alpha$ 出现在 P 上，那么由于 P 非矛盾，所以 $T\alpha$ 并不出现在 P 上，且 $\mathcal{V}(\alpha)=F$。

(ii)假设 $T(\alpha\wedge\beta)$ 出现在非矛盾路径 P 上。因为 τ 是完成表，所以 $T(\alpha)$ 和 $T(\beta)$ 都出现在 P 上。由归纳假设 $\mathcal{V}(\alpha)=T=\mathcal{V}(\beta)$，所以 $\mathcal{V}(\alpha\wedge\beta)=T$。

(iii)假设 $F(\alpha\wedge\beta)$ 出现在非矛盾路径 P 上。根据完成表的定义，$F\alpha$ 或 $F\beta$ 必出现在 P 上。由归纳假设知，无论是谁出现在 P 上，它都与 $\mathcal{V}$ 相一致。因此，要么 $\mathcal{V}(\alpha)=F$，要么 $\mathcal{V}(\beta)=F$。以上任何一种情况都能推出 $\mathcal{V}(\alpha\wedge\beta)=F$。

其余的联结词可以依照以上两种情形之一进行讨论，具体依照哪种情形取决于该联结词所对应的原子表是否分叉。细节证明留作习题 2。 □

证明(定理 5.3) 假设 α 永真，即对每个赋值 $\mathcal{V}$，$\mathcal{V}(\alpha)=T$。考虑以 $F\alpha$ 为根的任意完成表 τ。(由定理 4.8 知，以 $F\alpha$ 为根的 CST 即是完成表。)如果 τ 有一条非矛盾路径 P，那么由引理 5.2 知，存在一个赋值 $\mathcal{V}$，它与 P 上所有表值相一致，所以也与 $F\alpha$ 相一致。这就给出了一个赋值，使得 $\mathcal{V}(\alpha)=F$，与 α 的永真性相悖。因此，τ 中每条路径都是矛盾的，从而 τ 是 α 的一个表证明。 □

由完全性定理(实际上是引理 5.4)的证明，我们考虑构造 α 的一个表证明(即从 $F\alpha$ 开始)，如果尽最大努力所构造的以 $F\alpha$ 为根的完成表却不是 α 的证明(即完成表中至少有一条非矛盾路径)，那么根据引理 5.4，由此非矛盾路径定义的赋值给出了“α 永真”的一个反例。因为对任意给定的表值，我们总能得到以其为根的完成表，所以对每一个命题，我们都能找到它的一个表证明，或者找到关于其永真性的一个反例。

二分法(即便在更为复杂的谓词逻辑中)奠定了许多问题的可构造解基础。同时它也是 PROLOG 以及其他构造性定理证明器执行的理论基础。对给定的假设，比如“不存在 x 使得

$\mathcal{P}(x)$成立”，我们要么证明它是真的，要么找到一个反例，即找到一个 x，使得$\mathcal{P}(x)$成立。我们在第二章第五节考虑这些问题，更多的细节放在第三章中讨论。

习题

1. 验证引理 5.2 中其余的原子表情形。
2. 验证引理 5.4 中其余的联结词情形。

依照本节内容，表述并证明表反驳和可满足性。

3. 如果 α 是表可反驳的，即存在一个以 $T\alpha$ 为根的矛盾表，那么 α 是不可满足的(unsatisfiable)，即不存在赋值$\mathcal{V}$，使得$\mathcal{V}(\alpha)=T$。
4. 如果 α 是不可满足的，那么存在 α 的一个表反驳。

第六节 前件演绎和紧致性

回忆第三节最后关于命题(我们称之为前件)集合 Σ 的后承的讨论。命题 σ 是 Σ 的后承($\Sigma\models\sigma$)，如果 Σ 的任一模型(赋值)都是 σ 的模型，即使得 Σ 的全体元素为真的所有赋值都使 σ 为真。(见定义 3.2 和定义 3.8。)这里后承的概念以及与之相关的从一个前件出发的证明形式(即将给出定义)，反映了数学论证中的常见用法：数学定理通常表述为蕴涵形式 $\alpha\rightarrow\beta$；而定理的证明则表述为如下模式：首先假定前提假设(α)为真，然后证明结论(β)必为真。由证明的语法观点看，可以把该过程描述成“假定”α 成立，然后“演绎”出 β。后承的语义概念反映了第一种论证用法。现在我们要通过定义“从前件的集合出发证明某个命题”这一过程，来给出后承的语法版本或证明论版本。一旦我们引入了适当的概念，上述非形式化数学论证方法的形式化版本就很容易由演绎定理(习题 6)给出。为表述这一结果，现在给出所需概念的抽象陈述。

首先分析带有前件的表的定义，这里前件是从某个语句集合中取出。该定义与基本表的定义的区别在于，对前件 α，只能添加形如 $T\alpha$ 的表值。这一区别的直观反映即是，从一个前件集合出发就意味着假定了集合中的前件为真。

定义 6.1(带有前件的表) 令 Σ 为一个(可能无穷的)命题集合。归纳定义**带有 Σ 中前件的有穷表**(finite tableaux with premises from Σ)(或者简称为**从 Σ 出发的有穷表**(finite tableaux from Σ))：

(i)每个原子表都是从 Σ 出发的有穷表。

(ii)如果 τ 是从 Σ 出发的有穷表，且 $\alpha\in\Sigma$，那么 τ 通过添加 $T\alpha$ 到每条不包含 $T\alpha$ 的非矛盾路径的末端而得到的表也是从 Σ 出发的有穷表。

(iii)如果 τ 是从 Σ 出发的有穷表，P 是 τ 中一条路径，E 是出现在 P 上的一个表值，τ' 是 τ 通过添加以 E 为根的唯一的原子表到 P 的末端而得到的表，那么 τ'也是从 Σ 出发的有穷表。

如果 τ_0，τ_1，…，τ_n，…是一个从 Σ 出发的有穷表的(有穷或无穷)序列，使得对每个 $n\geqslant 0$，τ_{n+1}是由 τ_n 应用(ii)或(iii)得到的，那么 $\tau=\cup\tau_n$ 是**从 Σ 出发的表**(tableau from Σ)。

现在我们可以像之前那样来定义表证明。

定义 6.2 命题 α 的从 Σ 出发的表证明(tableau proof of a proposition α from Σ)是指，以 $F\alpha$ 为根表值的从 Σ 出发的表是矛盾的，即其上所有路径都是矛盾的。如果存在这样一个证明，则称 α 是**从 Σ 出发可证的**(provable from Σ)，记作 $\Sigma\vdash\alpha$。

例 6.3 图 15 给出了从前件集合$\{\neg B, (A \vee B)\}$出发的命题 A 的表证明。

现在我们可以模仿上一节来证明前件演绎的可靠性和完全性定理。仅有的变化在于完成表和 CST 的定义。从 Σ 出发的完成表(finished tableau from Σ)是指从 Σ 出发的表在定义 4.3 的意义下是完成的，并且对每个 $\alpha \in \Sigma$，$T\alpha$ 出现在所有非矛盾路径上。这里的主要思想仍是把前件的真值纳入到分析中来。

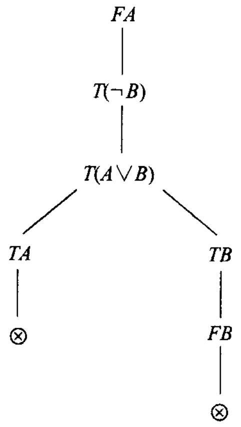


图 15

类似地，我们必须一步步地构造从 Σ 出发的 CST 来保证这些前件的出现。列举 Σ 的元素为 α_m，$m \in \mathcal{N}$，并通过只对 τ_{m+1} 的定义增加一步来修改 CST 的定义。如果新的构造已经生成 τ_m，那么令 τ'_{m+1}为按照标准 CST 过程定义出的下一个表。(如果该过程因所有路径都是矛盾的而终止，那么也终止当前的构造。)现在我们把 $T\alpha_m$ 添加到 τ'_{m+1}中所有不包含 $T\alpha$ 的非矛盾路径的末端，从而得到新的 τ_{m+1}。

定理 6.4 所有从前件集合出发的 CST 都是完成的。

证明 留作习题 1。 □

现在我们即可进行可靠性和完全性定理的证明，但有一点始终要注意：在相关赋值中，Σ 中的命题都为真。对一些必要的引理和定理我们只作陈述，大部分证明留作习题。

引理 6.5 如果赋值$\mathcal{V}$使所有 $\alpha \in \Sigma$ 为真，且与某个从 Σ 出发的表 τ 的根相一致，那么 τ 中存在一条路径使得其上所有表值与$\mathcal{V}$相一致。

证明 留作习题 2。 □

定理 6.6(前件演绎的可靠性) 如果存在 α 的从 Σ 出发的表证明，那么 α 是 Σ 的后承，即 $\Sigma \vdash \alpha \Rightarrow \Sigma \vDash \alpha$。

证明 若不然，则存在一个赋值，使得所有 $\beta \in \Sigma$ 为真而 α 为假。根据定理 5.1 的证明继续下去即得。 □

引理 6.7 令 P 是从 Σ 出发的完成表 τ 上的一条非矛盾路径。像引理 5.4 一样定义赋值 $\mathcal{V}$。$\mathcal{V}$与 P 上所有表值相一致，特别地，$\mathcal{V}$使得每个命题 $\beta \in \Sigma$ 为真(因为根据从 Σ 出发的完成表的定义，对所有的 $\beta \in \Sigma$，$T\beta$ 都出现在 P 上)。

证明 留作习题 3。 □

定理 6.8(前件演绎的完全性) 如果 α 是前件集合 Σ 的一个后承，那么存在 α 的一个从 Σ 出发的表演绎，即 $\Sigma \vDash \alpha \Rightarrow \Sigma \vdash \alpha$。

证明 如果 $\Sigma \vDash \alpha$，那么每个使得 Σ 中全部命题为真的赋值$\mathcal{V}$也使 α 为真。考虑以 $F\alpha$ 为根的从 Σ 出发的 CST。由定理 6.4 知，它是完成的。然后应用引理 6.7 即得。 □

这里仍然要考虑证明的有穷性问题。如果 Σ 是有穷的，那么经过与定理 4.11 相似的论证可以得出，从 Σ 出发的 CST 也是有穷的。(见习题 12。)对无穷的前件集合，论证过程类似于应用库尼西引理论证定理 4.9。

定理 6.9 如果 $\tau = \cup \tau_n$ 是从 Σ 出发的矛盾表，那么对某个 m，τ_m 是从 Σ 出发的有穷矛盾表。特别地，如果从 Σ 出发的 CST 是一个证明，那么它是有穷的。

证明 留作习题 4。 □

因此，如果 α 是从 Σ 出发可证的，那么存在 α 的一个有穷表证明。这可看作是紧致性

定理的语法版本。利用完全性和可靠性定理，我们可以将其转化为语义版本。

定理 6.10(紧致性)　*α 是 Σ 的后承，当且仅当 α 是 Σ 的某个有穷子集的后承。*

证明　留作习题 5。 □

我们已经把紧致性定理语义版本的间接证明(通过完全性和可靠性得到)留作习题。实际上，这一结果也可以直接证明。紧致性定理确实要比我们之前证过的定理深刻，因而也值得我们用两种方法来证明。直接证明法的优势在于，完全性定理可以从紧致性定理出发证明得到，且不依赖于无穷表。直接证明法还揭示了，紧致性定理其实只是库尼西引理的一个推论。

定义 6.11　*我们称命题集合 Σ 是* **可满足的**(satisfiable)，*如果它有一个模型，即存在一个赋值 $\mathcal{V}$，使得 $\mathcal{V}(\alpha)=T$ 对所有 $\alpha\in\Sigma$ 成立。我们也称该赋值* **满足**(satisfies)Σ。

例 6.12

(i) $\{A_1, A_2, (A_1\wedge A_2), A_3, (A_1\wedge A_3), A_4, (A_1\wedge A_4), \cdots\}$ 是一个可满足的无穷的命题集合。

(ii) $\{A_1, A_2, (A_1\rightarrow A_3), (\neg A_3)\}$ 是一个不可满足的有穷的命题集合，任何包含它的集合也都是不可满足的。

定理 6.13(紧致性)　*令 $\Sigma=\{\alpha_i \mid i\in\omega\}$ 为一个无穷的命题集合。Σ 是可满足的，当且仅当 Σ 的所有有穷子集 Γ 是可满足的。*

证明　注意，定理的必要性显然成立。另一方向则不然(不是平凡的)，原因在于，寻找不同的赋值来满足越来越长的初始段并不意味着存在某一个赋值满足整个序列。构造这样一个赋值，本质上就是库尼西引理的一个应用。

令 $\langle C_i \mid i\in\omega\rangle$ 为所有命题字母的一个串。定义树 T，其节点为二叉序列，按扩展序排列。用 $\mathrm{lth}(\sigma)$ 表示序列 σ 的长度，集合 $T=\{\sigma \mid$ 存在一个赋值 $\mathcal{V}$，使得对 $i\leqslant \mathrm{lth}(\sigma)$，$\mathcal{V}(\alpha_i)=T$ 且 $\mathcal{V}(C_i)=T$，当且仅当 $\sigma(i)=1\}$。这个定义要表述的思想是，我们只在一种情况下才把 σ 设置为树上的节点，即如果把 σ 解释成对命题字母 $C_i(i\leqslant \mathrm{lth}(\sigma))$ 的真值指派，则导致其中一个 $\alpha_i(i\leqslant \mathrm{lth}(\sigma))$ 为假。注意，如果 $\sigma\subseteq\tau\in T$，那么 $\sigma\in T$，从而 T 是二叉树。

断言：T 中有一条无穷路径，当且仅当 Σ 是可满足的。

断言的证明：如果 $\mathcal{V}$ 满足 Σ，那么由定义，使得 $\mathcal{V}(C_i)=T$ 当且仅当 $\sigma(i)=1$ 的所有 σ 构成的集合就是 T 上的一条路径。另一方面，假设 $\langle\sigma_j \mid j\in\mathcal{N}\rangle$ 是 T 上的一条无穷路径。令 $\mathcal{V}$ 是由取决于 σ_j 的真值指派扩张而成的唯一赋值，即该赋值使 C_i 为真当且仅当对某个 j，$\sigma_j(i)=1$(等价地，σ_i 是由扩展得到的线性序当且仅当对每个满足 $i\leqslant \mathrm{lth}(\sigma_j)$ 的 i，$\sigma_j(i)=1$)。如果 $\mathcal{V}\nvDash\Sigma$，那么存在某个 $\alpha_j\in\Sigma$，使得 $\mathcal{V}(\alpha_j)=F$。由推论 3.4 知，这取决于 $\mathcal{V}$ 对仅有的有穷多个命题字母指派真值。假设它仅取决于 $i\leqslant n$ 的那些 C_i，由 T 的定义易知，T 上不存在长度 $\geqslant n$ 的 σ。由于只存在有穷多个长度 $\leqslant n$ 的二叉序列 σ，这与假设"$\langle\sigma_j\rangle$ 是 T 上的一条无穷路径"相矛盾，所以 $\mathcal{V}\vDash\Sigma$。

下一个断言：对每一个 n，T 中都存在一个长度为 n 的 σ。根据假设，Σ 的所有有穷子集都是可满足的。因此，对每一个 n，存在一个赋值 $\mathcal{V}_n$，使得对每个 $i\leqslant n$，α_i 为真。根据定义，满足条件"$\sigma(i)=1$ 当且仅当 $\mathcal{V}_n(C_i)=T$，$i\leqslant n$"的 σ 在 T 上。

由库尼西引理(定理 1.4)知，T 中存在一条无穷路径，因此 Σ 是可满足的，得证。 □

习题 9 和 10 考察了紧致性定理的命题逻辑版本与拓扑版本之间的联系。习题 7 和 8 则

给出了紧致性定理的其他应用。

习题

1. 证明定理 6.4。

2. 证明引理 6.5。

3. 仿照引理 5.4 的证明过程证明引理 6.7。

4. 仿照定理 4.9 的证明过程证明定理 6.9。

5. 用前面的结论证明定理 6.10。

6. **演绎定理**(deduction theorem) 令 Σ 为一个有穷的命题集合，$\wedge\Sigma$ 为其元素的合取。证明对任意命题 α，以下各式相互等价：

(a)$\Sigma\vDash\alpha$

(b)$\vDash\wedge\Sigma\rightarrow\alpha$

(c)$\Sigma\vdash\alpha$

(d)$\vdash\wedge\Sigma\rightarrow\alpha$

紧致性的应用

对于习题 7 和 8，可以利用命题逻辑的紧致性定理或者库尼西引理。解决问题的关键在于忠实地把给定的问题转化成一个适当的命题集合(或命题树)，然后应用紧致性定理或者库尼西引理。最后必须再把应用结果还原成问题中的相应术语。谓词逻辑也考察了同样的几个问题，见第二章第七节习题 5。

7. 偏序的宽度至多为 n(width at most n)，如果每一个由两两不可比较的元素组成的集合，其大小都至多为 n。偏序 $<$ 中的链仅仅是指由 $<$ 中的某个线性序构成的子集。证明：宽度至多为 3 的无穷偏序可以分成 3 条链(不必互不相交)，如果每个宽度至多为 3 的有穷偏子序可以分成 3 条链。

提示(利用紧致性)：令偏序元素为 $\{p_n \mid n\in\mathcal{N}\}$。对 i，$j\in\mathcal{N}$，考虑命题 Rp_ip_j，Ap_i，Bp_i 和 Cp_i。把 Rp_ip_j 看作 $p_i<p_j$，而把 Ap_i 理解成 p_i 在链 A 上，Bp_i 和 Cp_i 与此类似。现在把要表述的结论写成命题集合的形式：A，B，C 都是链；偏序中每一个元素都在 A 或 B 或 C 中；偏序宽度为 3。

备注 Dilworth 定理描述了任意宽度至多为 n 的偏序可以分成 n 条链。因此，利用紧致性，无穷序的 Dilworth 定理可由有穷序的 Dilworth 定理推出。通过对给定序的大小作归纳，我们证明了有穷情形，所以对无穷情形的证明可视为紧致性的一个非平凡应用。

8. 图(graph)G 是由顶点集合 $\{a_0, a_1, \cdots\}$ 和边的集合 $\{a_i, a_j\}$ 组合而成。我们称 G 是 n-色的，如果我们可以用 C_1，$\cdots$，C_n 这 n 种颜色标识其顶点，使得 G 上任何一条边上的两个顶点都不相同。假设 G 的所有有穷子图(一些顶点构成的有穷子集以及其间的边)是 4-色的。证明：G 是 4-色的。

提示(利用库尼西引理)：定义一棵树，其节点是按扩展序排列的 4-元序列。我们把一个长度为 $n+1$ 的序列 σ 设置为树的节点，当且仅当它通过给 a_j 染色 $C_{\sigma(j)}$ 从而定义节点 a_0，a_1，$\cdots$，a_n 的一个 4-色。

备注 四色定理指出，所有平面图都是 4-色的。根据本题，我们需要证明有穷图的情形，因为一个图是平面图当且仅当它的所有有穷子图是平面图。

拓扑紧致性和库尼西引理的联系

对任意的命题集合 Σ，令形如 $\{\mathcal{V}\colon (\exists\alpha\in\Sigma)(\mathcal{V}\nvDash\alpha)\}$ 的集合所生成的集合为开集，$\mathcal{T}$ 是由这些开集决定的所有可能的真值赋值构成的集合。命题逻辑的紧致性定理可以和集合 $\mathcal{T}$ 上的拓扑联系起来。

9. 证明：具有如此拓扑的空间 $\mathcal{T}$ 是紧的。

10. 推导紧致性定理(6.13)语义版本的非平凡方向。(提示：利用开覆盖性质证明另一个方向。)

11. 从定理 6.13 出发证明库尼西引理。

有关库尼西引理和拓扑紧致性的其他联系，见第一节习题 10 和 11。

前件的有穷集合和表证明

12. 假定 Σ 是一个有穷的命题集合。证明：所有从 Σ 出发的 CST 都是有穷的。

*第七节 公理方法

命题演算(以及其他数学系统)经常表述为公理(axiom)与推理规则(rule of inference)的集合。命题逻辑的公理是某些永真命题。一般地，推理规则 R，由命题的某一 n-元组 α_1，…，α_n 出发“推理”出命题 α，并且期望在推理过程中保持永真性。因此，R 若要成为一个可接受的推理规则，它必须满足下述条件：如果我们可以从永真命题 α_1，…，α_n 出发，利用 R 推出 α，那么 α 也是永真的。

下面简要描述一个基于充分集 $\{\neg, \rightarrow\}$ 的经典构成。(为简单起见，我们把其余联结词视作由 $\neg$ 和 $\rightarrow$ 定义出来的。如此即有效地减少了所需公理的数目。)

公理 7.1 我们的系统公理由如下形式的命题组成：

(i) $(\alpha\rightarrow(\beta\rightarrow\alpha))$

(ii) $((\alpha\rightarrow(\beta\rightarrow\gamma))\rightarrow((\alpha\rightarrow\beta)\rightarrow(\alpha\rightarrow\gamma)))$

(iii) $(\neg\beta\rightarrow\neg\alpha)\rightarrow((\neg\beta\rightarrow\alpha)\rightarrow\beta)$

这里 α，β 和 γ 可以是任意命题。

上述命题形式通常称作公理模式(axiom scheme)。公理即是上述模式的所有实例，因为 α，β 和 γ 可以取遍所有命题。容易验证这些公理都是永真的。稍后我们会证明，在某种意义上，它们的选择是正确的。我们的系统中只有一条推理规则，即假言推理(modus ponens)。

规则 7.2(假言推理) 由 α 和 $\alpha\rightarrow\beta$，我们可以推出 β。该规则写作如下形式：

$$\begin{array}{c}\alpha\\ \underline{\alpha\rightarrow\beta}\\ \beta\end{array}$$

如上所示的基于公理和规则的系统通常叫做希尔伯特式证明系统。因此我们在此系统下用 $\vdash_H$ 表示可证性。

定义 7.3 令 Σ 为一个命题集合。

(i) 一个**从 Σ 出发的证明**(proof from Σ)是指一个由 α_1，α_2，…，α_n 构成的有穷序列，使得对每个 $i\leq n$，以下三条之一成立：

(1) α_i 是 Σ 中的元素；

(2) α_i 是公理；

(3) α_i 可以通过应用推理规则，由之前的某些 α_j 推出。

(ii) α 是**从 Σ 出发可证的**(provable from Σ)，记作 $\Sigma\vdash_H\alpha$，如果存在一个从 Σ 出发的证明 α_1，…，α_n，这里 $\alpha_n=\alpha$。

(iii) α 的**证明**(proof)仅仅是指从 $\varnothing$ 出发的证明；α 是**可证的**(provable)，如果从 $\varnothing$ 出发，它是可证的。

例 7.4 这里给出 $((\neg\beta\rightarrow\alpha)\rightarrow\beta)$ 的一个从 $\Sigma=\{\neg\alpha\}$ 出发的证明：

$\neg\alpha$	从 Σ 出发
$(\neg\alpha\rightarrow(\neg\beta\rightarrow\neg\alpha))$	公理(i)
$(\neg\beta\rightarrow\neg\alpha)$	假言推理
$((\neg\beta\rightarrow\neg\alpha)\rightarrow((\neg\beta\rightarrow\alpha)\rightarrow\beta))$	公理(iv)
$((\neg\beta\rightarrow\alpha)\rightarrow\beta)$	假言推理

注意，正如表演绎的情形，尽管前件集合 Σ 可能是无穷的，但是，如果 α 是从 Σ 出发可证的，那么 α 是从 Σ 的某个有穷子集出发可证的。证明过程总是有穷的。

标准的定理仍是可靠性、完全性和紧致性定理。可靠性相当容易证明，只需验证所有公理永真且推理规则(假言推理)保持永真，即如果对某个赋值前件为真，则结论也为真。紧致性定理的语法版本显而易见，因为所有证明都是有穷的。语义版本(如定理 6.13 所述)仍是非平凡的。当然，定理 6.13 的语义证明在这里仍然可用。对此基于规则的系统，紧致性定理也可以通过完全性定理推导出来(因此也一定是非平凡的)。

对此特殊系统，我们略去可靠性和完全性的证明(参见 Mendelson [1979, 3.2])，但在下一节考虑另外一个基于规则的系统时，我们会给出证明。现在，只给出本系统相关定理的表述。

定理 7.5(从前件出发的可靠性和完全性)　*α 是从命题集合 Σ 出发可证的，当且仅当 α 是 Σ 的后承，即 $\Sigma\vdash_H\alpha\Leftrightarrow\Sigma\vDash\alpha$。*

推论 7.6(可靠性和完全性)　*命题 α 是可证的，当且仅当它是永真的，即 $\vdash_H\alpha\Leftrightarrow\ \vDash\alpha$。*

评注 7.7

(i) *关于假言推理*(on modus ponens)：如果 α 和 $\alpha\to\beta$ 分别有一个表证明，那么根据可靠性定理，α 和 $\alpha\to\beta$ 都是永真的。由于假言推理保持永真性，所以 β 也是永真的。因此，根据表证明的完全性定理，β 有一个表证明。事实上，存在一个算法，它从 α 和 $\alpha\to\beta$ 的表证明出发得到 β 的表证明，此即 Gentzen Hauptsatz(原则定理)。证明过长，此处略去。假言推理也叫*截规则*(cut rule)，因此该定理也叫*截消定理*(cut elimination theorem)。

(ii) *关于定理*(on theorem)：*定理*是指任何可证的命题。因此，作为一个元素出现在证明过程中的任何命题都是定理。我们通常把结论理解成整个证明过程的最后一环，其实证明过程的任意一个初始段也是证明。

(iii) *公理的选择*(choice of axiom)：推论 7.6 说明，在下述意义下公理是完全的：从公理出发，我们可以通过反复应用假言推理来证明任何永真命题。另一方面，因为公理是永真的且假言推理保持永真性，所以每个定理(即本系统中的可证命题)都有一个表证明。因此，表证明是充分的，公理以及上述的推理规则也是充分的。公理可以再多一些(或少一些)，推理规则也可以再多一些(或换成其他推理规则)，或者公理和推理规则同时发生改变。具体的选择有时取决于个人喜好，有时则为了方便(例如，为了证明过程简单)。关键在于，无论是怎样的证明系统，定理集合都只有一个，即所有永真命题构成的集合。

(iv) *效率*(efficiency)：从上述公理系统和规则出发高效地证明定理是需要技巧的，因为我们不得不经常猜测要用哪条公理，而不像表证明那样有一个系统的程序。这一问题的提出是由一类包含过多备选公理的公理系统引起的。这里给出的希尔伯特式的证明系统就是由很多的公理和极少的规则构成的。其他一些与之相反(公理少规则多)的公理系统有甘岑(Gentzen)系统和自然演绎系统，它们与自动定理证明的联系更紧密。从这些公理系统的直观或者构造上讲，它们更多地与生成具有下述性质的系统联系起来：对任意给定的命题，能够给出其证明或者给出一个反例(如第五节最后所述)。

第八节 消 解

PROLOG 以及大多数自动定理证明器的证明方法都是消解(resolution)，消解是一个由公理和规则构成的特别简单高效的系统。正如第七节所示的系统，消解只有一个规则，它通过消去所有公理来生成证明，从而减少了大量的猜测工作。(事实上，它通过各种构成规则来自动地组合各条公理，然而就生成证明的工作量而言，这与公理消去基本相当。)正如我们对于表方法的描述，消解方法也是一个反驳过程，即它试图证明给定的公式是不可满足的。首先假定公式是合取范式(见第三节习题 3 与第四节习题 10 ~ 11)。在经典的计算机科学术语当中，这种形式叫做子句形式(clausal form)，其他与之相关的术语如下：

定义 8.1

(i)**文字**(literal)ℓ 是指命题字母 p 或其否定 $\neg p$。如果 ℓ 是 p 或 $\neg p$，那么 $\bar{\ell}$ 分别表示 $\neg p$ 或 p。命题字母也叫做**正文字**(positive literal)，它们的否定叫做**负文字**(negative literal)。

(ii)**子句**(clause)C 是指文字的有穷集合(应当视作文字元素的析取)。我们说 C 为真，当且仅当 C 中有一个元素为真。由此，**空子句**(empty clause)$\square$ 始终为假，因为它没有真值为真的元素。

(iii)**公式**(formula)S 是指子句的(不必有穷)集合(应当视作子句元素的合取)。我们说 S 为真，当且仅当 S 中每一个元素为真。由此，**空公式**(empty formula)$\varnothing$ 始终为真，因为它没有真值为假的元素。

(iv)**指派**(assignment)$\mathcal{A}$ 是指一些文字的相容集，即对任意命题字母 p，该集合不同时包含 p 和 $\neg p$。(显然，这只是一个(部分)真值指派，对 $p \in \mathcal{A}$ 的 p 指派为 T，而对 $\bar{q} \in \mathcal{A}$ 的 q 指派为 F。)**完全指派**(complete assignment)是指对每个命题字母 p，该指派包含 p 或 $\neg p$。它对应于定义 3.1 中的真值指派。

(v)$\mathcal{A}$**满足**(satisfies)S，记作 $\mathcal{A} \vDash S$，当且仅当 $\forall C \in S$，$C \cap \mathcal{A} \neq \varnothing$，即由 $\mathcal{A}$ 导出的赋值使得 S 中所有子句为真。

(vi)公式 S 是**(不)可满足的**，如果(不)存在满足它的指派 $\mathcal{A}$。

例 8.2

(i)p，q，r，$\neg p$，$\bar{q}$($=\neg q$)，$\bar{r}$ 以及 $\neg\bar{q}$($=q$)都是文字。

(ii)$\{p, r\}$，$\{\neg q\}$ 以及 $\{q, \neg r\}$ 都是子句。

(iii)$S=\{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{s, \neg t\}\}$ 是一个公式，在我们原来的表示系统中，应记作 $((p \vee r) \wedge (q \vee \neg r) \wedge (\neg q) \wedge (\neg p \vee t) \wedge (s \vee \neg t))$。

(iv)如果 $\mathcal{A}$ 由 $\{p, q, r, s, t\}$ 给出，即 $\mathcal{A}$ 是使得 $\mathcal{A}(p)=T=\mathcal{A}(q)=\mathcal{A}(r)=\mathcal{A}(s)=\mathcal{A}(t)$ 成立的(部分)指派，那么 $\mathcal{A}$ 是不满足(iii)中公式 S 的指派。但是 S 是可满足的。

* PROLOG 的记法

考虑子句形式或者合取范式的另一个角度就是推理。假设子句 C 的正文字(包含于 C 的命题字母)为 $A_1, \cdots, A_m$，负文字(使得 $\bar{p}$(即 $\neg p$)在 C 中的那些命题字母 p 的否定)为 $B_1, \cdots, B_n$。那么子句 C 等价于 $A_1 \vee A_2 \vee \cdots \vee A_m \vee \neg B_1 \vee \cdots \vee \neg B_n$，它又等价于 $B_1 \wedge B_2 \cdots \wedge B_n \to A_1 \vee \cdots \vee A_m$。如果 C 中至多只有一个正文字(即至多一个 A_i)，那么 C 叫做 Horn 子句(Horn clause)(如果恰好只有一个正文字，那它叫做程序子句(program clause))。如果 Horn 子句包

含负文字，那它叫做规则(rule)，否则就叫做事实(fact)。目标子句(goal clause)是指不含正文字的子句。PROLOG 的核心正是逻辑学和涉及 Horn 子句(将在第十节分析)的证明论。(然而 PROLOG 并不局限于命题字母，它还允许文字中出现变元。我们将在第二章谓词逻辑中详细说明这一点。)

在 PROLOG 的标准表示中，我们采用与→相反的符号←或:-，读作"如果"。符号∧的出现被逗号所代替。因此，A_1:-B_1，B_2，…，B_n 或者 $A_1 \leftarrow B_1$，…，B_n 读作(并且意味着)A_1 如果(B_1 且 B_2 且……且 B_n)。在生成演绎或写程序时，我们把诸如 A_1:-B_1，B_2，…，B_n 的子句 C 当作指定条件，在此条件下 A_1 为真。我们通常对证实某个结论感兴趣。因此，A_1 叫做子句 C 的目标(goal)(有时叫做 C 的头(head))，B_1，…，B_n 叫做 C 的子目标(subgoal)(或者体(body)，在此术语系统下符号:-叫做颈(neck))。C 告诉我们，为了证实 A，首先应该证实每个 B_1，…，B_n。在"目标-子目标"术语系统下，考虑术语"成功"和"失败"。目标 A 成功，是指 A 为真，或者更准确地从程序设计的观点来说，是指存在 A 的一个证明。反之我们说目标失败。不过需要注意的是，成功和失败这一术语(至少到目前为止)是不太精确的。

对符号:-的退化情形，即当 $n=0$ 或 $m=0$ 时，我们考虑它们分别代表的含义。如果 $m=0$，那么:-B_1，…，B_n(或者 $\leftarrow B_1$，…，B_n)叫做目标子句(goal clause)，它等价于 $\neg B_1 \vee \cdots \vee \neg B_n$，即它断言 B_i 之中有一个文字失败(为假)。如果 $n=0$，那么 A_1:-(或者 $A_1 \leftarrow$)叫做单元子句(unit clause)，它等价于 A_1；因此该符号只是说 A_1 成功(为真)。

消解规则(resolution rule)很像假言推理的截规则版本。假言推理(见第七节)说从 α 和 $\alpha \to \beta$ 出发可以推出 β。按照同样的模式，截规则说从 $\alpha \vee \gamma$ 和 $\neg\alpha \vee \beta$ 出发可以推出 $\gamma \vee \beta$。因此，截规则要比假言推理更一般，因为它允许额外命题 γ 的参与。消解是截规则的限制版本，在消解中，α 必须是文字，而 β 和 γ 必须是子句。

定义 8.3(消解) 采用当前术语，如果分别从形如 $\{\ell\} \sqcup C_1'$，$\{\bar{\ell}\} \sqcup C_2'$ 的子句 C_1，C_2 出发，推导出 $C = C_1' \cup C_2'$，那么 C 叫做 C_1，C_2 的**消解式**(resolvent)。(这里 ℓ 是任意的文字，$\sqcup$ 表示不相交集合的并。)我们也把 C_1，C_2 称作**父子句**(parent)，把 C 称作它们的**儿子句**(child)，然后说我们**对**(文字)ℓ **进行了消解**(resolved on ℓ)。

注意，与截规则的经典形式相比，消解规则也消去了冗余，即 C_1 和 C_2 中相同的文字。该规则取代了经典证明系统(比如第七节的希尔伯特式系统)中的某些公理。

显然，消解是一种可靠的规则，即它对所有真值指派保持可满足性。如果某个指派同时满足 C_1，C_2，那么无论它对要消解的文字 ℓ 指派何值，它都必然满足 C_1' 和 C_2' 其中之一，因此它满足消解式 $C_1' \cup C_2'$。(形式化的论证见引理 8.12。)因此，消解规则可作为证明可靠性的基础。

定义 8.4 **C 的从给定公式 S 出发的(消解)演绎或证明**((resolution) deduction or proof of C from a given formula S)是指子句的一个有穷序列 C_1，C_2，…，$C_n = C$，其中每个 C_i 要么是 S 中的成员，要么是子句 C_j，C_k(j，$k<i$)的消解式。如果存在这样一个演绎，则称 C 是**从 S 出发(消解)可证的**((resolution) provable from S)，记作 $S \vdash_{\mathcal{R}} C$。□的从 S 出发的演绎叫做 **S 的(消解)反驳**((resolution) refutation)。如果存在这样一个演绎，则称 S 是**(消解)可反驳的**((resolution) refutable)，记作 $S \vdash_{\mathcal{R}} \Box$。

提醒 S 的消解反驳给出了□的一个从 S 出发的证明。由于□始终为假，因此，把 S 的消解反驳理解成证明 S 不可能为真，即 S 是不可满足的。这就是可靠性定理(定理 8.11)。

例 8.5

(i)从$\{p, r\}$和$\{\neg q, \neg r\}$出发，(对 r 进行)消解得到$\{p, \neg q\}$。

(ii)从$\{p, q, \neg r, s\}$和$\{\neg p, q, r, t\}$出发，(对 p 或 r 进行)消解分别得到$\{q, \neg r, s, r, t\}$或$\{p, q, s, \neg p, t\}$。显然，这两个消解式都是永真的，等价于空公式。

消解证明的演绎树形式比上述的序列形式更有用。

定义 8.6 C 的从 S 出发的**消解树证明**(resolution tree proof of C from S)是指具有下列性质的有穷标签二叉树 T:

(i)T 的根标识为 C。

(ii)T 的叶子标识为 S 中的元素。

(iii)如果任意非叶子节点 σ 标识为 C_2，其直接后继 σ_0，σ_1 分别标识为 C_0，C_1，那么 C_2 是 C_0 和 C_1 的消解式。

例 8.7 图 16 给出了公式 $S=\{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{\neg s\}, \{s, \neg t\}\}$ 的消解树反驳，即□的从 S 出发的消解树证明：

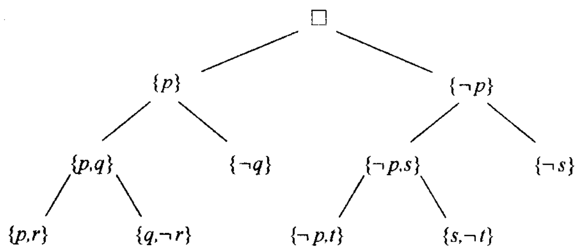


图 16

引理 8.8 C 有一个从 S 出发的消解树证明，当且仅当存在 C 的从 S 出发的消解演绎。

证明 ($\Rightarrow$)按照任一与序 $<$ 相反的顺序，列出 C 的从 S 出发的树证明中所有节点的标签(因此，首先列出叶子，最后才列出根)。这个序列可以看作 C 的从 S 出发的消解演绎，验证消解演绎的定义即可。

($\Leftarrow$)我们对 C 的从 S 出发的消解演绎的长度作归纳。假设对任何长度 $<n$ 的演绎，我们可以得到相应的树证明。同时假设 $C_1, \cdots, C_n$ 是从 S 出发的一个长度为 n 的消解演绎。如果 $C_n \in S$，则不证自明。否则，C_n 是 C_i，$C_j(i, j<n)$的消解式。由归纳，C_i，C_j 分别对应着树证明 T_i，T_j。令树 T_n 是以 C 为根标签，且以 T_i，T_j 为直接后继。根据定义，T_n 即是我们要找的树证明。 □

对于从 S 出发可证的那些定理或子句的集合，其归纳定义还对应了消解演绎的另外一种表述。

定义 8.9 $\mathcal{R}(S)$是 S **在消解下的闭包**(closure of S under resolution)，即由下述归纳定义所决定的集合：

1. 如果 $C\in S$，则 $C\in\mathcal{R}(S)$。
2. 如果 C_1，$C_2\in\mathcal{R}(S)$且 C 是 C_1 和 C_2 的消解式，则 $C\in\mathcal{R}(S)$。

命题 8.10 对任意子句 C 和公式 S，存在 C 的一个从 S 出发的消解演绎，当且仅当 $C\in\mathcal{R}(S)$。特别地，存在 S 的一个消解反驳，当且仅当□$\in\mathcal{R}(S)$。

证明 留作习题 1。 □

首先要注意的是，无论消解方法如何表述，它都能提供一个可靠的证明过程。

定理 8.11(消解的可靠性) 如果存在 S 的一个消解反驳，则 S 是不可满足的。

首先证明一个引理，定理证明的归纳步会用到它。

引理 8.12 如果公式(即子句的集合) $S=\{C_1, C_2\}$ 是可满足的，且 C 是 C_1 和 C_2 的一个消解式，那么 C 是可满足的。事实上，任何满足 S 的指派 $\mathcal{A}$ 都满足 C。

证明 由于 C 是 C_1 和 C_2 的一个消解式，所以存在 ℓ，C_1' 和 C_2'，使得 $C_1=\{\ell\}\sqcup C_1'$，$C_2=\{\bar{\ell}\}\sqcup C_2'$ 以及 $C=C_1'\cup C_2'$。因为 $\mathcal{A}$ 满足 $\{C_1, C_2\}$，所以它满足(即包含) C_1 和 C_2。又因为 $\mathcal{A}$ 是一个指派，所以不会出现下述情形：$\ell\in\mathcal{A}$ 且 $\bar{\ell}\in\mathcal{A}$。不妨设 $\bar{\ell}\notin\mathcal{A}$。因为 $\mathcal{A}\vDash C_2$ 且 $\bar{\ell}\notin\mathcal{A}$，所以 $\mathcal{A}\vDash C_2'$，因此 $\mathcal{A}\vDash C$。关于 $\ell\notin\mathcal{A}$ 的情形，只需把 C_2 换成 C_1，$\bar{\ell}$ 换成 ℓ 即可。 □

证明(定理 8.11) 如果 $C_1, \cdots, C_n$ 是从 S 出发的消解演绎，那么上面的引理对 n 作归纳，证明了任何满足 S 的指派满足所有的 C_i。如果该演绎是 S 的反驳，那么 $C_n=\square$。由于没有指派能够满足 □，所以 S 是不可满足的。 □

评注 8.13 可靠性定理及其证明也可从定义 8.6 或定义 8.9 出发直接给出。我们把这种表述留作习题 2 和 3。

下面的主要目标是证明消解方法的完全性，即如果 S 是不可满足的，则存在 S 的一个消解反驳。自然地，我们要考虑搜索 S 的反驳的执行方法。首先考虑利用先前的消解方法，然后引入消解的更多限制版本，这些新的消解方法在保持可靠性和完全性的前提下让搜索变得更加高效。由此思路，我们首先给出定义 8.3 中一般形式消解的完全性之简单直接的证明。不过这个证明依赖于紧致性定理(的语义版本)。然后引入并分析不可满足性的一种抽象描述。它提供了消解演绎的完全性定理的另一个证明，该证明不依赖紧致性定理，它同时也提供了紧致性定理的一个新的证明。新的消解完全性证明为第九节中加细消解完全性的证明提供了一种模式。

新的证明从下面的引理开始，该引理允许我们消去下述子句中的文字：这些子句可以从不可满足公式 S 出发消解演绎得到。反复应用该引理，从而证明：不含任何文字的子句 □ 可以从 S 出发消解演绎得到。

定理 8.14 对任意公式 T 和文字 ℓ，令 $T(\ell)=\{C\in\mathcal{R}(T)\mid \ell, \bar{\ell}\notin C\}$。如果 T 不可满足，则 $T(\ell)$ 也不可满足。

证明 假设 T 不可满足，为了导出矛盾，假设 $\mathcal{A}$ 是满足 $T(\ell)$ 的任意指派，$\mathcal{A}$ 定义在除 ℓ 之外的所有(T 的)文字上。令 $\mathcal{A}_1=\mathcal{A}\cup\{\ell\}$，$\mathcal{A}_2=\mathcal{A}\cup\{\bar{\ell}\}$。由于 T 不可满足，所以存在 T 中子句 C_1，C_2，使得 $\mathcal{A}_1\nvDash C_1$，$\mathcal{A}_2\nvDash C_2$。现在由于 $\ell\in\mathcal{A}_1$，$\mathcal{A}_1\nvDash C_1$，所以 $\ell\notin C_1$。如果 $\bar{\ell}$ 也不在 C_1 中，那么由定义知 $C_1\in T(\ell)$。而这与假设 $\mathcal{A}\vDash T(\ell)$ 相矛盾，因此 $\bar{\ell}\in C_1$。同理，$\ell\in C_2$。因此，我们可以对 C_1，C_2 消解 ℓ 得到子句 D。D 不包含 ℓ，因此 D 在 $T(\ell)$ 中。(作为 T 中两个子句的消解式，D 显然在 $\mathcal{R}(T)$ 中。)于是，根据选择的指派 $\mathcal{A}$，$\mathcal{A}\vDash D$。然而，如果 $\mathcal{A}$ 满足消解式 D，那么它必须满足 C_1，C_2 其中之一。由此，得出矛盾。 □

定理 8.15(消解的完全性) 如果 S 不可满足，则存在 S 的一个消解反驳。

证明 根据紧致性定理(定理6.13)，存在 S 的一个有穷子集 S' 是不可满足的。由于任何从 S' 出发的反驳演绎也是从 S 出发的反驳演绎，所以可以假设 S 是有穷的，即只包含有穷多个子句。如果 S 中只有有穷多个子句并且每个子句都是有穷的，那么 S 的任何子句中都只有有穷多个文字，不妨设为 ℓ_1，ℓ_2，…，ℓ_n。在下面的证明中我们仅考虑基于这 n 个文字的子句和公式。

我们希望考察子句 $C\in\mathcal{R}(S)$ 的集合，并证明该集合包含□。通过应用引理8.14我们可以逐一消去每个文字，下面从 $S_n=S(\ell_n)=\{C\in\mathcal{R}(S)\mid \ell_n,\ \bar{\ell}_n\notin C\}$ 开始。根据定义，S_n 是 S 的不包含 ℓ_n 和 $\bar{\ell}_n$ 的消解后承的集合。由引理8.14知，S_n 是不可满足的。然后令 $S_{n-1}=S_n(\ell_{n-1})$，它是 S_n(因而也是 S)的不包含 ℓ_{n-1}，$\bar{\ell}_{n-1}$，ℓ_n 和 $\bar{\ell}_n$ 的消解后承的集合，该集合是不可满足的。照此法继续进行，可以定义 S_{n-2}，…，S_0。反复应用该定义以及引理8.14，可以看到 S_0 是 S 的不包含任何文字的消解后承的集合，该集合是不可满足的。因为不包含任何文字的公式只有∅和{□}，∅又是可满足的，所以□ $\in S_0$。因此，□是 S 的消解后承，得证。 □

下面我们研究关于消解演绎之完全性证明中出现的概念和引理的更为抽象的表述。这种表述在第九节和第十节处理加细消解时会用到。

定义8.16 *如果 S 是公式，ℓ 是文字，令*

$$S^{\ell}=\{C-\{\bar{\ell}\}\mid C\in S\wedge \ell\notin C\}$$

因此，S^{ℓ} 是由 S 中不包含 ℓ 与 $\bar{\ell}$ 的那些子句 C 以及使得 $C\cup\{\bar{\ell}\}\in S$ 的(不包含 ℓ 的)那些子句共同组成。注意，如果单元子句 $\{\bar{\ell}\}$ 在 S 中，那么□在 S^{ℓ} 中。

坦白地说，这个定义初读起来有些晦涩。它的主要思想就是逐层分解 S(的可满足性)。S^{ℓ} 对应了假设 ℓ 为真时分解的结果，而 $S^{\bar{\ell}}$ 对应了假设 ℓ 为假时分解的结果。例如，考虑假设 ℓ 为真时的公式 S。首先要注意的是，如果 ℓ 为真，那么包含 ℓ 的任何子句都可满足，因为子句等价于其文字的析取。由于公式 S 等价于其子句的合取，所以任何已知为真的子句都可以从 S 中消去而不改变 S 的可满足性。因此，假设 ℓ 为真，可以从 S 中略去所有包含 ℓ 的子句并且保持 S 的可满足性。这正是 S^{ℓ} 的定义要说明的：把要考虑的子句限制到那些 C 上，使得 $\ell\notin C$。其次要注意的是，仍然假设 ℓ 为真，$\bar{\ell}$ 可以从包含它的任意子句 C 中略去而不改变 C 的可满足性。(C 等价于其文字的析取。如果其中一个文字已知为假，那么它不再影响整个析取式的可满足性。)当然，如果 C 的可满足性不受影响，那么对包含 C 的公式 S，其可满足性也不受影响。这也是 S^{ℓ} 的定义要说明的：用更短的子句 $C-\{\bar{\ell}\}$ 来替换 C。

如果 ℓ 为假，则 $\bar{\ell}$ 为真，对 $S^{\bar{\ell}}$ 应用上述分解过程。由于 ℓ 与 $\bar{\ell}$ 之中必有一个为真，所以可以断言(正如在引理8.19中所做的) S 是可满足的，当且仅当 S^{ℓ} 和 $S^{\bar{\ell}}$ 其中之一是可满足的。因此，可以将 S 的可满足性问题归约为两个与之相似但是简化了的问题，即少掉一个命题字母的 S^{ℓ} 和 $S^{\bar{\ell}}$ 的可满足性问题。于是继续进行分解，逐一考虑这两个新的公式 S^{ℓ} 和 $S^{\bar{\ell}}$。利用这种方法，可以生成一棵以公式为节点的二叉树，且在该树的每一层都能消去一个文字。该树的每一条路径都对应着一个指派。经过 S^{ℓ} 的分支使得 ℓ 为真，而经过 $S^{\bar{\ell}}$ 的分支使得 ℓ 为假。如果树上所有路径都以一个包含空子句□的公式为终结，那么可以得知原公式 S 是不可满足的。另一方面，如果并非所有路径导出□，那么，如果成功地消去所有出现在 S

中的文字，则要么存在一条无穷路径，就是沿着它我们消去了所有文字，要么至少存在一条路径是以空公式$\varnothing$为终结。在以上两种情况下，S 都是可满足的。事实上，那条适当的(无穷的或导出$\varnothing$的)路径直接提供了一个满足 S 的指派。

这种分解方法的基本框架与以 $F\alpha$ 开始的表证明相似，这里 α 是某个命题。在表证明中，我们也尝试分析所有使 α 为假的路径，即验证 $F\alpha$。如果所有路径都导出矛盾($\otimes$)，可以得知 $F\alpha$ 是不可满足的，α 永真。这里，如果所有路径都导出一个包含不可满足子句$\square$的公式，则公式 S 是不可满足的。另一方面，如果表分解是完成的并且生成了一条非矛盾路径，那么我们可以利用该路径(引理 5.4)来定义一个满足 α 的赋值。在由定义 8.16 引出的分解方法中，当我们消去所有文字(对应着表的完成)时，只剩下一条无穷路径或一条以空公式$\varnothing$为终结的路径，那么该路径本身即直接提供了一个满足 S 的指派。

我们通过以下两个例子来陈述如何从 S 出发构造 S^{ℓ}，并给出这种分析方法的一般形式。

例 8.17 令 $S=\{\{p\},\{\neg q\},\{\neg p,\neg q\}\}$。首先消去 p，然后消去 q，分解过程由图 17中的树给出：

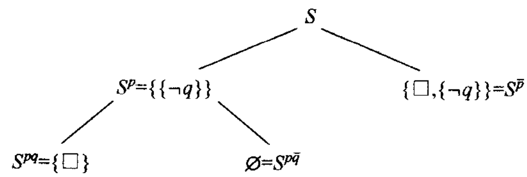


图 17

假设 p 为真，从 S 中消去子句$\{p\}$，从子句$\{\neg p,\neg q\}$中消去文字$\neg p$，从而在树的第一层的左边得到 S^{p}。假设 p 为假，右边($S^{\bar p}$)就约化为$\{\square,\{\neg q\}\}$，因为$\{p\}$为 S 中的一个子句，S 断言 p 为真。在下一层，考虑 q。在左边，如果假设 q 为真，我们又得到$\square$，因为 S^{p} 断言$\neg q$为真。在右边，假设 q 为假，我们消去所有包含$\neg q$的子句，从而得到空公式。因此，我们得到了一条以$\varnothing$为终结的路径。该路径提供了满足 S 的指派：使 p 为真，q 为假。

例 8.18 考虑例 8.7 中被证明为不可满足的公式 $S=\{\{p,r\},\{q,\neg r\},\{\neg q\},\{\neg p,t\},\{\neg s\},\{s,\neg t\}\}$。分解的第一步就是消去 p。如果假设 p 为真，消去包含 p 的子句(因为它们为真)，同时消去其余子句中的$\neg p$($\neg p$ 为假，对满足这些子句不起作用)，从而得到 $S^{p}=\{\{q,\neg r\},\{\neg q\},\{t\},\{\neg s\},\{s,\neg t\}\}$。另一方面，如果假设 p 为假，消去包含$\neg p$的子句，同时消去其余子句中的 p，从而得到 $S^{\bar p}=\{\{r\},\{q,\neg r\},\{\neg q\},\{\neg s\},\{s,\neg t\}\}$。图 18 给出了整个树分解的一部分。

经过 S^{pq}分解的路径在该点终结，因为该点包含$\square$，所以是不可满足的。然而另外一条路径则可以继续进行，并且如果继续下去的话，所有路径最终都会以一个包含$\square$的不可满足公式为终结。这与 $S\vdash_{\mathcal{R}}\square$的证明相似。我们把本例的完整分解留作习题 4。

现在我们表述并证明，上述分解准确地阐释了可满足性的概念。

引理 8.19 S 可满足，当且仅当 S^{ℓ} 和 $S^{\bar\ell}$ 其中之一可满足。(提醒：从左往右证时，不必取用同一个指派。)

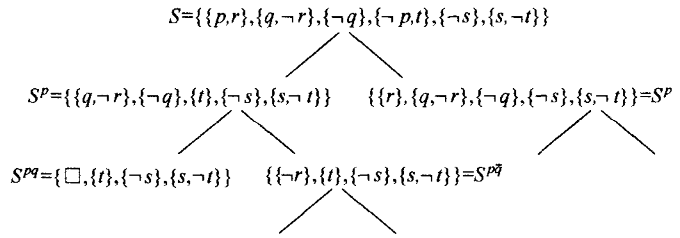


图 18

证明 ($\Rightarrow$)假设$\mathcal{A}\vDash S$。如果$\mathcal{A}$是一个完全指派，那么，它必定使 ℓ，$\bar{\ell}$之一为真。不妨设 ℓ 为真。于是可以证明$\mathcal{A}\vDash S^{\ell}$。如果不愿假设$\mathcal{A}\vDash S$，那么可以从如下事实出发：根据定义，$\ell$ 或$\bar{\ell}$之中必有一个不属于$\mathcal{A}$。为明确起见，假设$\bar{\ell}\notin\mathcal{A}$。现在同样可以断言$\mathcal{A}\vDash S^{\ell}$。我们必须证明$\mathcal{A}$满足 S^{ℓ} 中的所有子句。考虑任意的 $C\in S^{\ell}$。根据 S^{ℓ} 的定义，要么 $C\cup\{\bar{\ell}\}\in S$，要么 $C\in S$(取决于把 C“放”到 S^{ℓ} 里的 S 中的那个子句是否包含$\bar{\ell}$)。因此，根据假设，$\mathcal{A}\vDash C$ 或者$\mathcal{A}\vDash C\cup\{\bar{\ell}\}$。由于只有当指派包含子句中的某个文字时，它才满足该子句，所以存在一个文字 k，使得要么 $k\in C\cap\mathcal{A}$，要么 $k\in(C\cup\{\bar{\ell}\})\cap\mathcal{A}$。根据假设，$\bar{\ell}\notin\mathcal{A}$，所以在两种情形下都有 $k\in C\cap\mathcal{A}$，即$\mathcal{A}\vDash C$。$\ell\notin\mathcal{A}$的情形可以类似地处理。

($\Leftarrow$)为明确起见，假设$\mathcal{A}\vDash S^{\ell}$。现在在 S^{ℓ} 的任何子句中都不出现 ℓ 和$\bar{\ell}$，因此可以利用 ℓ 来调整$\mathcal{A}$而不影响 S^{ℓ} 的可满足性。更准确地说，如果令$\mathcal{A}'=(\mathcal{A}-\{\bar{\ell}\})\cup\{\ell\}$，那么$\mathcal{A}'\vDash S^{\ell}$。我们断言$\mathcal{A}'\vDash S$。考虑任意的 $C\in S$。如果 $\ell\in C$，因为 $\ell\in\mathcal{A}'$，所以$\mathcal{A}'\vDash C$。如果 $\ell\notin C$，根据 S^{ℓ} 的定义，$C-\{\bar{\ell}\}\in S^{\ell}$。由于$\mathcal{A}\vDash S^{\ell}$，则存在某个文字 $k\in(C-\{\bar{\ell}\})\cap\mathcal{A}$。现在$\mathcal{A}$和$\mathcal{A}'$的差别至多在于 ℓ 和$\bar{\ell}$。由于 $k\neq\ell$ 或$\bar{\ell}$，因而得到 $k\in\mathcal{A}'\cap C$。 □

推论 8.20 *S 不可满足，当且仅当 S^{ℓ} 和 $S^{\bar{\ell}}$ 不可满足。*

该推论和□的不可满足性一起，从本质上刻画了不可满足性的性质。

定理 8.21 *如果 UNSAT $=\{S\mid S$ 是不可满足的公式$\}$，那么 UNSAT 即是由下列子句归纳定义出的公式的集合$\mathcal{U}$:*

(i)$\square\in S\Rightarrow S\in\mathcal{U}$

(ii)$S^{\ell}\in\mathcal{U}\wedge S^{\bar{\ell}}\in\mathcal{U}\Rightarrow S\in\mathcal{U}$

证明 因为□不可满足，所以 UNSAT 满足(i)。根据推论 8.20，它也满足(ii)。因此，$\mathcal{U}\subseteq$UNSAT。我们必须证明 UNSAT$\subseteq\mathcal{U}$。用反证法：如果 $S\notin\mathcal{U}$，则 S 是可满足的。令$\{p_i\}$为满足下述条件的命题字母的集合：p_i 或$\bar{p}_i$ 出现在 S 的某个子句中。归纳定义序列$\{\ell_i\}$，使得 $\ell_i=p_i$ 或$\bar{p}_i$，且 $S^{\ell_1,\cdots,\ell_i}\notin\mathcal{U}$。(性质(ii)保证了我们总能找到这样的 ℓ_i。)现在令$\mathcal{A}=\{\ell_i\mid i\in\mathcal{N}\}$。我们断言$\mathcal{A}$满足 S。假设 $C\in S$。我们必须证明 $C\cap\mathcal{A}\neq\varnothing$。因为 C 是有穷的，所以存在一个 n，使得对所有出现在 C 中的命题字母 p_i，都有 $i<n$。如果 $C\cap\mathcal{A}=\varnothing$，则 $\forall i<n(\ell_i\notin C)$，因此对每一步 $S^{\ell_1,\cdots,\ell_i}(i<n)$，对应于 C 的子句都要发生变动。在这样的每一步变动中，比如 $S^{\ell_1,\cdots,\ell_i}$，我们把$\bar{\ell}_i$ 从子句中消去。由于 C 中所有文字都取自$\bar{\ell}_i$，从 C 出发经过 $S^{\ell_1,\cdots,\ell_n}$推导出的子句为□。根据选择的 ℓ_i，$S^{\ell_1,\cdots,\ell_n}\notin\mathcal{U}$。另一方面，由(i)知任何包含□的 S 都在$\mathcal{U}$中，从而导出矛盾。 □

该结论与引理5.4相似。序列ℓ_i的选择对应了引理5.4中指派的定义，标号命题字母出现在完成表的非矛盾路径上。类似地，我们构造一个指派，使其满足正在构造中的路径上的所有表值。我们在此构造中最终到达不可满足子句□，从而导出矛盾。对表证明而言，上面这个关于不可满足性的刻画的确是消解方法完全性证明的核心。

定理8.22(消解方法的完全性) *如果S不可满足，则存在S的一个消解反驳(等价地，$\Box \in \mathcal{R}(S)$)。*

证明 我们根据定理8.21给出的UNSAT的刻画来作归纳证明。显然，如果$\Box \in S$，则$\Box \in \mathcal{R}(S)$。对归纳步，假设对某个ℓ和S，$\Box \in \mathcal{R}(S^{\ell})$且$\Box \in \mathcal{R}(S^{\bar{\ell}})$。我们必须证明$\Box \in \mathcal{R}(S)$。根据假设，我们有□的分别从$S^{\ell}$和$S^{\bar{\ell}}$出发的树证明$T_0$和$T_1$。考虑$T_0$。如果$T_0$中每个叶子都被$S$中的子句所标识，那么$T_0$即是□的从$S$出发的证明。否则定义树$T_0'$如下：对$T_0$中标签子句不在$S$中的那些叶子，将其上(包括该叶子)的所有节点标签$C$改成$C \cup \{\bar{\ell}\}$。我们断言：$T_0'$是$\{\bar{\ell}\}$的从$S$出发的证明树。显然，根据$S^{\ell}$的定义，$T_0'$的所有叶子都在$S$中。现在我们必须验证每个$T_0'$的非叶子节点都是被其直接后继$C_0'$和$C_1'$的消解式$C'$所标识。假设$C'$，$C_0'$,$C_1'$在$T_0$中分别对应子句$C$，$C_0$，$C_1$。因为$T_0$是消解树证明，所以$C$是$C_0$和$C_1$的消解式。首先注意，$T_0$中没有对$\ell$或$\bar{\ell}$进行的消解，因为$T_0$上的任何标签都不含$\ell$或$\bar{\ell}$(由$S^{\ell}$的定义得知)。然后考虑$T_0'$上子句$C_0'$，$C_1'$和$C'$的可能形式。例如，如果$C_0$，$C_1$(因此$C$)都在标签子句不在$S$中的那些叶子之上，那么$C' = C \cup \{\bar{\ell}\}$是$C_0' = C_0 \cup \{\bar{\ell}\}$和$C_1' = C_1 \cup \{\bar{\ell}\}$的消解式，因此$T_0'$即是我们要找的消解树证明。下面再考虑其余情形：要么三个子句在T_0'和T_0中完全一样，要么添加$\{\bar{\ell}\}$到C_0，C_1其中之一以及C上。在以上两种情形中，C'仍是C_0'和C_1'的消解式，同样可以验证T_0'是消解树证明。类似地，对T_1中标签子句不在S中的那些叶子，如果将其上(包括该叶子)的所有节点标签C改成$C \cup \{\ell\}$，我们会得到$\{\ell\}$的从S出发的树证明T_1'(或者，如果所有叶标签在S中，得到的将是□的树证明)。现在可以定义□的从S出发的树证明T如下：T的根标签为□，根的两个直接后继分别为T_0'和T_1'。由于□是$\{\ell\}$和$\{\bar{\ell}\}$的消解式，所以T是□的从S出发的树证明。 □

***紧致性再研究**

显然没有必要重新证明紧致性定理，因为在语义意义下，它的证明可以看作是唯一的。不过下面给出的证明是基于定理8.21对UNSAT作出的刻画。定理8.21的证明中构造的无穷文字序列ℓ_i，正好对应了定理6.13关于紧致性的最初证明中指派树上的路径(由库尼西引理给出)。

定理8.23(紧致性) *如果S是不可满足的，则S的某个有穷子集是不可满足的。*

证明 令$\mathcal{T} = \{S \mid \exists S_1 \subseteq S[S_1$有穷且不可满足$]\}$。如果能够证明$\mathcal{T}$满足定理8.21的(i)(ii)两条，那么它就包含了所有不可满足的公式，从而完成证明。

(i)如果$\Box \in S$，则$S_1 = \{\Box\} \subseteq S$，$S \in \mathcal{T}$得证。

(ii)假设S^{ℓ}，$S^{\bar{\ell}} \in \mathcal{T}$。我们必须证明$S \in \mathcal{T}$。根据$\mathcal{T}$，$S^{\ell}$和$S^{\bar{\ell}}$的定义，存在有穷公式$S_1$，$S_2 \subseteq S$，使得$S_1^{\ell} \subseteq S^{\ell}$，$S_2^{\bar{\ell}} \subseteq S^{\bar{\ell}}$且$S_1^{\ell}$，$S_2^{\bar{\ell}}$均是不可满足的。令$S_3 = S_1 \cup S_2$。$S_3$是$S$的一个有穷子集。我们需要证明它是不可满足的。如果$S_3$是可满足的，则存在一个指派$\mathcal{A}$满足它。现在$\mathcal{A}$必须省去$\ell$或$\bar{\ell}$。因此，$\mathcal{A}$满足$S_3^{\bar{\ell}}$或$S_3^{\ell}$，于是它满足$S_2^{\bar{\ell}}$或$S_1^{\ell}$(因为$S_3 \supset S_2$，$S_1$)，从而得到矛盾。 □

习题

1. 用归纳法证明命题 8.10。(提示：其中一个方向对证明的行数作归纳，另一个方向对$\mathcal{R}(S)$的定义作归纳。)
2. 根据消解树证明以及对其定义作归纳，重述定理 8.11(可靠性)的证明。
3. 从由$\Box \in \mathcal{R}(S)$定义的消解演绎出发，重述定理 8.11(可靠性)的证明。
4. 继续分解例 8.18，直至所有路径都以某个与不可满足子句□等价的公式作为终结。
5. 把下列公式改写成合取范式以及子句形式。
   (a)$((A \vee B) \to (C \vee D))$
   (b)$\neg(A \wedge B \wedge \neg C)$
   (c)$\neg((A \wedge B) \vee (B \vee C) \vee (A \wedge C))$
6. 下面哪个子句集合可满足？如果可满足，给出满足它的指派。如果不可满足，解释原因。
   (a)$\{\{A, B\}, \{\neg A, \neg B\}, \{\neg A, B\}\}$
   (b)$\{\{\neg A\}, \{A, \neg B\}, \{B\}\}$
   (c)$\{\{A\}, \Box\}$
   (d)$\{\Box\}$
7. 找出下列子句对的所有消解式：
   (a)$\{A, B\}, \{\neg A, \neg B\}$
   (b)$\{A, \neg B\}, \{B, C, D\}$
8. 找出下列集合 S 的$\mathcal{R}(S)$：
   (a)$\{\{A, \neg B\}, \{A, B\}, \{\neg A\}\}$
   (b)$\{\{A\}, \{B\}, \{A, B\}\}$
9. 从下面的公式出发，找出空子句的一个演绎：
   (a)$\{\{A, \neg B, C\}, \{B, C\}, \{\neg A, C\}, \{B, \neg C\}, \{\neg B\}\}$
10. 利用消解证明下列公式对任何指派都不可满足：
   (a)$(A \leftrightarrow (B \to C)) \wedge ((A \leftrightarrow B) \wedge (A \leftrightarrow \neg C))$
   (b)$\neg(((A \to B) \to \neg B) \to \neg B)$
11. 令 α 为命题$\neg(p \vee q) \to (\neg p \wedge \neg q)$。
   (a)给出 α 的表证明。
   (b)把$\neg\alpha$ 转化成 CNF 以及子句形式。(给出转化步骤。)
   (c)给出 α 的消解证明。
12. 按照习题 11 的模式处理命题$\beta = (\neg r \vee (p \wedge q)) \to ((r \to p) \wedge (r \to q))$。
13. 证明：如果 $S \vdash_{\mathcal{R}} C$，则 $S \vDash C$。
14. 证明：如果 $S \cup \{\neg A\} \in$ UNSAT，则 $S \vdash_{\mathcal{R}} A$。
15. 令$\mathcal{T}$是由下列子句归纳定义而来：
   (a)$\{\Box\} \in \mathcal{T}$
   (b)$S^{\ell}, S^{\bar{\ell}} \in \mathcal{T} \Rightarrow S \in \mathcal{T}$

   证明：对任何有穷的 $S \in$ UNSAT，$S \in \mathcal{T}$但并非所有的 $S \in$ UNSAT 都在$\mathcal{T}$中。(因此，在定理 8.21 对 UNSAT 的刻画中，基本步假设囊括了所有包含□的公式，不能把它改成只针对空公式□作假设。)

第九节　加细消解

对于由第七节的规则和公理组成的经典系统，消解在效率上已经得到了相当的提高。从直观上看，消解理应更加高效，因为我们无须过问证明的下一步需要(第七节)哪些公理(公

理总数是无穷多的)。消解只有一条规则。因此,当我们试图系统地搜索给定(有穷)的 S 的消解反驳时,只需要验证这一规则在 S 中的子句和先前演绎出的子句当中的应用。即便如此,搜索空间也会迅速增大。事实上,对于某一类定理,标准的消解方法需要花费指数时间。因此,我们的主要任务就是研究限制搜索空间的方法(最好能保持方法的可靠性和完全性,尽管在实际应用中两者时常都不能保持;关于这一点,稍后会详述)。坦白地说,限制证明的搜索空间意味着损失一些证明。因此,虽然搜索的空间变小了,但是我们找到的证明却要比原搜索空间下更长了。总的来说,修剪搜索树后效率的确提高了。(显然,我们是在启发式的意义下谈论效率。在复杂性理论的意义下,$\mathrm{SAT}=\{S \mid S$ 可满足$\}$ 是 NP-完全的(见 Garey, Johnson[1979, 5.3]),没有系统可以超出这个理论极限。但是在实际应用中,搜索空间越小,运行时间越短。)对于把搜索引向消解反驳的众多可能的策略,我们只考虑其中一小部分。

我们可以通过两点来引导搜索。第一点是终结无望路径上的搜索。第二点是通过指定搜索在分叉点的序来引导它。最明显需要修剪的分支可能就是那些带有重言式的分支:如果 C 是重言式,那么它对于说明 S 的不可满足性毫无帮助。由于验证子句 C 是否重言式很容易(只需验证对某个命题字母 p, p 与 $\bar{p}$ 同时出现其中),所以这样的修剪花费不多但却有用。(我们是在子句形式的条件下计算验证重言式的花费,然而把任意命题转化成 CNF 的花费可能会很大。)

定义 9.1 ***T*-消解**(T-resolution)是指父子句均非重言式的消解。$\mathcal{R}^T(S)$ 是 S 在 T-消解下的闭包。

引理 9.2 可靠的消解方法的任何限制版本,即允许更少的演绎,都是可靠的。特别地,由于消解是可靠的,所以 $\mathcal{R}^T$ 也是可靠的,即若 $\Box \in \mathcal{R}^T(S)$,则 S 不可满足。

证明 由于限制版本中的任何演绎都是原来消解系统中的演绎,而由可靠性知,原来系统中不存在 $\Box$ 的演绎,因此在限制版本中也不存在 $\Box$ 的演绎。 $\Box$

同时不难看出 $\mathcal{R}^T$ 是完全的。

定理 9.3(***T*-消解的完全性**) 如果 S 不可满足,则 $\Box \in \mathcal{R}^T(S)$。

证明 定理 8.22 中关于消解完全性的证明对 $\mathcal{R}^T$ 仍然适用。唯一需要注意的是,如果 T_0, T_1 中均无重言式,那么分别通过添加 $\bar{\ell}$, ℓ 到适当的子句而得到的树 T_0', T_1' 也无重言式。由于 $T_0(T_1)$ 是从 $S^{\ell}(S^{\bar{\ell}})$ 出发的证明,根据假设知,$T_0(T_1)$ 中所有子句都不包含 $\ell(\bar{\ell})$。 $\Box$

重言式在所有指派下均为真,因而可以忽略。我们还可以通过下述办法来强化这种语义方法,从而加细消解:固定一个指派 $\mathcal{A}$ 并且要求在每一步消解中都有一个子句在 $\mathcal{A}$ 的指派下为假。(如果消解的父子句在 $\mathcal{A}$ 下都为真,那么消解式也为真。如果没有在 $\mathcal{A}$ 下失败的子句,我们就不能期望得到不可解性。显然,这与我们可以忽略所有这般消解的证明相距甚远。)

定义 9.4 令 $\mathcal{A}$ 为一个指派。**$\mathcal{A}$-消解**($\mathcal{A}$-resolution)是指父子句之中至少有一个在 $\mathcal{A}$ 下为假的消解。$\mathcal{R}^{\mathcal{A}}(S)$ 是 S 在 $\mathcal{A}$-消解下的闭包。这个过程通常叫做**语义消解**(semantic resolution)。

定理 9.5（$\mathcal{A}$-消解的完全性） 对任意的$\mathcal{A}$和 S，如果 $S \in$ UNSAT，则$\Box \in \mathcal{R}^{\mathcal{A}}(S)$。

证明 固定一个指派$\mathcal{A}$，令$\mathcal{T}^{\mathcal{A}} = \{S \mid \Box \in \mathcal{R}^{\mathcal{A}}(S)\}$。我们必须证明 UNSAT$\subseteq \mathcal{T}^{\mathcal{A}}$。根据定理 8.21 关于 UNSAT 的刻画，我们只须证明

(i)$\Box \in S \Rightarrow S \in \mathcal{T}^{\mathcal{A}}$。

(ii)对任意的 S 和 ℓ，如果 $S^{\ell} \in \mathcal{T}^{\mathcal{A}}$且 $S^{\bar{\ell}} \in \mathcal{T}^{\mathcal{A}}$，那么 $S \in \mathcal{T}^{\mathcal{A}}$。

(i)显然成立。对(ii)考虑$\Box$的分别从 S^{ℓ} 和 $S^{\bar{\ell}}$ 出发的$\mathcal{A}$-消解证明 T_0 和 T_1。我们可以像定理 9.3 的证明那样添加$\bar{\ell}(\ell)$到 $T_0(T_1)$中适当的子句，从而形成 $T_0'(T_1')$。显然，这两棵树分别是$\{\bar{\ell}\}$和$\{\ell\}$的消解证明（也可能是$\Box$的消解证明）。然而它们不一定是$\mathcal{A}$-消解证明，因为$\bar{\ell}$，ℓ之一在$\mathcal{A}$下可能为真。另一方面，因为$\bar{\ell}$，ℓ中至多有一个在$\mathcal{A}$下为真，所以 T_0'和T_1'中至少有一个是$\mathcal{A}$-消解证明。为明确起见，设 $\ell \notin \mathcal{A}$，于是 T_1'是$\{\ell\}$或$\Box$的一个从 S 出发的$\mathcal{A}$-消解证明。对于后一种情形，我们已然得证。对于前者，我们可以把$\{\ell\}$的这个证明和 T_0 结合起来，从而得到我们要找的$\Box$的$\mathcal{A}$-消解证明如下：对 T_0 的每个不在 S 中的叶子 C，将其作为 $C \cup \{\bar{\ell}\}$与$\{\ell\}$的儿子句。因为 $\ell \notin \mathcal{A}$，所以这是一个$\mathcal{A}$-消解。由于 $C \notin S$，则 $C \cup \{\bar{\ell}\}$在 S 中。因此，除非$\{\ell\}$不在 S 中，我们就得到了$\Box$的从 S 出发的$\mathcal{A}$-消解证明。要完成整个证明的构造，我们还需把树 T_1'添加到每个以$\{\ell\}$为标签的叶子下面。现在容易看出，如此所得的树描述了$\Box$的一个从 S 出发的$\mathcal{A}$-消解演绎。和我们之前所考虑的形如 $C \cup \{\bar{\ell}\}$的节点与$\{\ell\}$之间的那些消解不同的是，新的证明中的所有消解都出现在$\mathcal{A}$-消解演绎树 T_0 或者 T_1'中。因此，所有出现在新的证明树上的消解都是$\mathcal{A}$-消解。 □

下面考虑*有序消解*(ordered resolution)，它是下述语法程序的一个例子，该程序至少在某种程度上决定了我们首先应该进行哪个消解。

定义 9.6 假设我们已经给所有的命题字母编了索引。对有序消解，我们像通常所做的那样来定义$\mathcal{R}^{<}(S)$，特殊的地方在于，当 p 的索引号高于 C_1，C_2 中任意命题字母的索引号时，只允许 $C_1 \sqcup \{p\}$和 $C_2 \sqcup \{\bar{p}\}$进行消解。

如果我们仅仅试图复原 p 和$\bar{p}$到$\Box$的从 S^p 和 $S^{\bar{p}}$ 出发的有序证明 T_0，T_1，进而模仿定理 8.22关于完全性的证明，那么我们得到的可能不再是有序消解。不过我们只需要重查关于 UNSAT 的刻画，以确定可以加上序。

定理 9.7 UNSAT 等价于由下列子句归纳定义出来的公式$\mathcal{U}^{<}$的类：

(i)$\Box \in S \Rightarrow S \in \mathcal{U}^{<}$。

(ii$^{<}$)如果没有命题字母的索引号严格小于出现在 S 中的 p 的索引号，$S^p \in \mathcal{U}^{<}$且 $S^{\bar{p}} \in \mathcal{U}^{<}$，那么 $S \in \mathcal{U}^{<}$。

证明 因为归纳子句(ii$^{<}$)要比 8.21 的(ii)弱，所以$\mathcal{U}^{<}$必定包含在$\mathcal{U}$ = UNSAT 中。另一方面，如果按索引号的升序排列出现在 S 中的$\{p_i\}$，那么关于 UNSAT 的刻画（定理 8.21）的最初证明表明，任何 $S \notin \mathcal{U}^{<}$均可满足，因而 UNSAT 也包含在$\mathcal{U}^{<}$中。 □

对于定理 8.22 中消解完全性的证明，如果把$\mathcal{R}$替换成$\mathcal{R}^{<}$，把其中的(ii)替换成(ii$^{<}$)，即可证明有序消解的完全性。

定理 9.8（有序消解的完全性） 如果 S 是不可满足的，则存在 S 的一个有序消解反驳，即$\Box \in \mathcal{R}^{<}(S)$。

要消解的文字的不同排列可能会导致相同的消解式，有序消解消除了诸如此类的一些重复。因此，它减少了导出任何特定子句所需的步数。除了上述的几种消解方法，还有很多其他的消解版本，每一个版本消去了搜索空间的某一个方面。习题当中将会讨论其中两个。此外，线性消解作为最高效的消解方法，将在下一节以及下一章的完全谓词逻辑中讨论。

习题

1. 假设 S 是子句的集合，$U\subseteq S$ 且 $S-U$ 可满足。我们说消解有一个*支集*(support) U，如果父子句并不同时在 $S-U$ 中。请给出带有支集 U 的子句的消解之完整定义以及相关集合 $\mathcal{R}^{U}(S)$ 的定义。证明：$S\in$ UNSAT $\Leftrightarrow$ $\Box\in\mathcal{R}^{U}(S)$。
2. 通俗地说，F-消解(F-resolution)是指子句之中有一个是目标子句(即该子句仅包含负文字)的消解。给出 F-消解的一个完整的形式定义(不要参照消解的基本定义)，同时定义 $S\vdash_F\Box$(存在□的一个从 S 出发的 F-消解证明)。证明：$S\in$ UNSAT，当且仅当 $S\vdash_F\Box$。
3. 令 S 是一个有穷的子句集合。对 S 中子句中的某个文字，任意给它的每次出现(occurrence)一个不重复的索引号。*锁定消解*(lock resolution)是指要消解的文字在每个父子句中都有(在该子句的所有文字当中)最低的索引号。儿子句中的文字继承它们在父子句中的索引号，如果某个文字在两个父子句中均有出现，那么在儿子句中该文字取其在父子句的较小的索引号。(我们用上标表示索引号。)
   例子：
   $$C_1=\{p^1,q^2,r^3\}$$
   $$C_2=\{\neg p^4,q^5\}$$
   $$C_3=\{\neg q^6\}$$
   $$S=\{C_1,C_2,C_3\}。$$
   这里可以通过锁定消解 C_1 和 C_2 得到 $\{q^2,\ r^3\}=C_4$。C_4 又可以和 C_3 进行锁定消解得到 $\{r^3\}$。不过不能锁定消解 C_2 和 C_3，因为对 C_2 和 C_3，我们只能消解 q，而 q 在 C_2 中的出现并不具有 C_2 中任何文字的最低索引号。(它的索引号是 5，而 $\neg p$ 的索引号是 4。)
   证明：锁定消解是完全的，即如果 S 不可满足，则存在□的一个从 S 出发的锁定消解演绎。(提示：对溢出文字数目(excess literal parameter)作归纳，这里溢出文字数目 $=S$ 中所有文字的出现总数 $-S$ 中子句的数目。)
4. 证明：锁定消解与重言式缺省结合在一起，不可能得到一个完全的消解系统。

第十节 线性消解、Horn 子句和 PROLOG

本节将考虑另外一种加细消解：线性消解。关于该法的全面分析，我们将在谓词逻辑一章中讨论。这里仅给出它的表述，分析它在 Horn 子句下的特殊情形。正是在此形式下线性消解成为了 PROLOG 基本定理证明器。我们期望通过一个消解的线性序列而非分叉树来作定理证明。我们完成了一个消解的序列，是指其中(从第二个消解开始)每一个消解的父子句中有一个是之前已经完成的某个消解的儿子句。

定义 10.1

*(i) C 的从 S 出发的***线性***(linear)(消解)演绎或证明是指满足 $C=C_{n+1}$ 以及下列条件的有序对序列 $\langle C_0,\ B_0\rangle$，…，$\langle C_n,\ B_n\rangle$：*

(1) C_0 以及每个 B_i，要么是 S 中的元素，要么是某个 $j<i$ 的 C_j。

(2) 每个 $C_{i+1}(i\leq n)$，是 C_i 和 B_i 的消解式。

*(ii) 通常我们说 C 是从 S 出发***线性可演绎的***(linearly deducible)或***线性可证的***，记作 $S\vdash_{\mathcal{L}}C$，如果存在 C 的一个从 S 出发的线性演绎。存在 S 的一个***线性反驳***(linear refutation)，*

如果 $S\vdash_{\mathcal{L}}\square$。$\mathcal{L}(S)$是由从 S 出发线性可演绎的所有子句构成的集合。

习惯上，线性消解的起始点写在顶端，结论写在底端。(这与树消解的图表相反，在那里标识结论的节点写在顶端。)由此我们画出一个线性消解的图表，如图 19 所示。

例 10.2 令 $S=\{A_1, A_2, A_3, A_4\}$，$A_1=\{p, q\}$，$A_2=\{p, \neg q\}$，$A_3=\{\neg p, q\}$，$A_4=\{\neg p, \neg q\}$。图 20 给出了 S 的一个线性反驳。

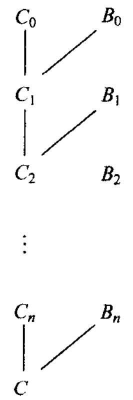


图 19

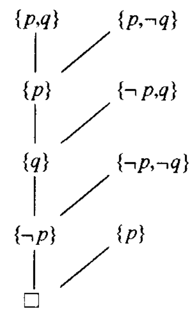


图 20

定义 10.3 在有关线性消解的行文中，考虑从其出发生成演绎的集合 S，我们常常把 S 中的元素称作**插入子句**(input clause)。C_i 称作**中间子句**(center clause)，B_i 称作**边子句**(side clause)。C_0 称作演绎的**开始子句**(starting clause)。

考虑扩展"父-儿"术语系统，如果在从 S 出发的 C 的消解证明中把子句 C 的祖先(ancestor)定义为出现在证明树中 C 以上的那些子句，那么可以将线性演绎重新定义如下：每个 C_i 与一个插入子句或者与 C_i 的某个祖先进行消解得到 C_{i+1}。

线性消解显然是一种加细消解；即每个线性消解证明都可以很容易地转化成一般的消解证明。由于消解是可靠的(定理 8.11)，所以线性消解也是可靠的。在第二章第十四节，我们将证明线性消解是完全的。现在我们只考虑 Horn 子句与 PROLOG 程序的情形。

定义 10.4

(i)**Horn 子句**是指至多包含一个正文字的非空子句。

(ii)**程序子句**是指恰好包含一个正文字的子句。(在 PROLOG 中，它表示成 $A{:-}B_1, B_2, \cdots, B_n$。)

(iii)如果程序子句中包含负文字，那么该子句叫做**规则**(即(ii)中 $n>0$ 的情形)。

(iv)**事实**(或叫**单元子句**)是指恰好由一个正文字组成的子句(记作 A 或 $A{:-}$)。

(v)**目标子句**是指不包含正文字的子句。(因此，在 PROLOG 中它是以问题的形式出现，记作 ?-。)

(vi)PROLOG **程序**(program)是指仅包含程序子句(规则或事实)的一个子句集合。

注意，Horn 子句要么是程序子句，要么是目标子句；而程序子句要么是规则，要么是事实。值得注意的是，只有把目标子句与事实放在一起才会出现不相容的情况。矛盾也许是由规则引发的，但是单独的规则(或者事实)不会生成矛盾。

引理 10.5 如果 Horn 子句的集合 S 是不可满足的，那么 S 必定至少包含一个事实和一个目标子句。

证明 使得所有命题字母为真的指派满足所有程序子句，而使得所有命题字母为假的指派满足所有目标子句和所有规则。因此，任何由 Horn 子句组成的不可满足的集合都必定同时包含一个事实和一个目标子句。 □

PROLOG 程序的一般思想如下：给定一个由事实和规则构成的集合，我们期望演绎出该集合的后承。通常地，我们希望知道事实 q_1，q_2，…，q_n 的合取是否可由程序 P 推出。在 PROLOG 提示中输入问题? - q_1，q_2，…，q_n，然后得到一个回复，它告诉我们 q_i 是否为程序 P 的后承。PROLOG 的基本执行思想是：添加目标子句 $G=\{\neg q_1, \neg q_2, \cdots, \neg q_n\}$ 到给定的程序，然后询问由 Horn 子句组成的集合 $P\cup\{G\}$ 是否不可满足。这里，我们需要注意一个简单但很重要的事实：q_1，q_2，…，q_n 的合取只有在 $P\cup\{G\}$ 是不可满足的情况下才是程序 P 的后承。我们把上面的语义表述单独写成一个引理。在 PROLOG 中，我们每次询问问题都会潜在地用到它。

引理 10.6 如果 P 是 PROLOG 程序，$G=\{\neg q_1, \neg q_2, \cdots, \neg q_n\}$ 是目标子句，那么所有的 q_i 都是 P 的后承，当且仅当 $P\cup\{G\}$ 不可满足。

证明 本证明仅是复述定义。首先注意 $P\cup\{G\}$ 不可满足，当且仅当任何满足 P 的指派使 G 为假。其次注意目标子句 G 为假，当且仅当所有的$\neg q_i$ 为假，即 G 为假当且仅当所有 q_i 为真。因此，我们所要找的事实的合取就是程序 P 在 $P\cup\{G\}$ 不可满足时的一个后承。 □

现在我们的目标就是把这个语义条件转化成能够被消解方法所验证的证明论中的条件。事实上我们证明了：对由 Horn 子句构成的那些集合，线性消解需要判定其不可满足性。

定理 10.7(关于 Horn 子句的线性消解的完全性) 如果 S 是由 Horn 子句构成的一个不可满足的集合，那么存在□的一个从 S 出发的线性消解演绎，即$\square\in\mathcal{L}(S)$。

证明 由紧致性定理(定理 6.13 或定理 8.23)，假设 S 是有穷的。下面对 S 中的文字数目作归纳。由引理 10.5 可知，至少存在一个正文字 p 是在 S 中作为事实$\{p\}$出现的。考虑如定义 8.16 所述的公式 S^p。S^p 中每个子句都是 S 中某子句的一个子集，故由定义知其为 Horn 子句。我们断言 S^p 不可满足。这里需要注意的是，如果$\mathcal{A}\vDash S^p$，则$\mathcal{A}\cup\{p\}\vDash S$，这与 S 的不可满足性相矛盾。由于 S^p 包含的文字比 S 少(我们略去了任何包含 p 的子句，并从剩余子句中消去了$\bar{p}$)，所以我们可以对 S^p 应用归纳假设，从而得到□的一个从 S^p 出发的线性消解演绎。定理 8.22 给出了一般消解方法的完全性证明，与该证明的归纳步一样，上面得到的线性消解演绎要么已经是□的从 S 出发的线性证明，要么可以将其转化成$\{\bar{p}\}$的从 S 出发的线性证明：对某个不在 S 中的子句，添加$\bar{p}$至其以下所有子句。现在我们可以对该证明进行一步扩展，即添加新的边子句$\{p\}\in S$，使其与最后一个中间子句$\{\bar{p}\}$进行消解，从而得到□，得证。 □

线性消解的优势非常明显。我们现在只需寻找一个线性序列来证明不可满足性，而无须找一整棵树。PROLOG 中搜索的树结构是根据边子句的各种可能性生成的。PROLOG 下所有可能演绎组成的树中，每条路径代表一个线性消解。给定一个 PROLOG 程序和一个目标子句(询问编译器的问题)，我们可以更精确地指定所要寻找的线性消解中各子句的消解顺序。

由引理 10.4 可知，在该演绎中目标子句必定会被用到。事实上，我们可以要求□的演绎从目标子句开始，并且边子句仅从 PROLOG 程序的子句中挑选。由于这些子句叫做插入子句，因此该消解限制版本叫做线性插入消解(linear input resolution)。

定义 10.8 令 P 是程序子句的集合，G 是一个目标子句。$S=P\cup\{G\}$ 的**线性插入**(linear input)(LI)**消解反驳**是指 S 的由 G 开始的线性消解反驳，并且该反驳中所有边子句均来自 P(插入子句)。

一般来说，LI-消解方法不是完全的。这一点由下面的例子可以看出。

例 10.9 回顾例 10.2 中的子句：$S=\{A_1, A_2, A_3, A_4\}$，$A_1=\{p, q\}$，$A_2=\{p, \neg q\}$，$A_3=\{\neg p, q\}$，$A_4=\{\neg p, \neg q\}$。这里唯一的目标子句是 A_4，令 $G=A_4$。然而剩余的子句并不全是程序子句。如果令 $P=\{A_1, A_2, A_3\}$，并试图生成 $S=P\cup\{G\}$ 的一个由 G 开始的线性插入消解反驳，那么结果只有失败。图 21 给出了一个尝试。

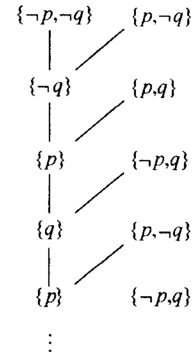


图 21

问题的关键在于，无论怎样开始这个消解，当我们遇到恰好包含一个文字的中间子句时，它与 P 中子句进行消解而生成的任何消解式仍然恰好包含一个文字。因此，我们永远不能推出□。

然而在 PROLOG 程序设计中，线性插入消解的确是完全的。

定理 10.10 令 P 是程序子句的集合，G 是一个目标子句。如果 $S=P\cup\{G\}\in$ UNSAT，则存在 S 的一个线性插入消解反驳。

证明 首先注意，目标子句只能与程序子句(对偶于另一个目标子句)进行消解，因为对于正在消解的一对子句，必然存在某个文字 p 出现于其中一个子句，$\bar{p}$ 出现于另一个子句，而目标子句仅包含负文字。此外，对于这种形式的消解，其儿子句必定也是目标子句。这是因为程序子句中的单个正文字是我们消解的唯一选择，在儿子句中该文字已被消去，只剩下程序子句中的负文字以及目标子句中的剩余文字。因此，对□的任意一个由 G 开始从 S 出发的线性证明，该证明中每一步消解的儿子句都必然是目标子句，并且其中所有的边子句都必然是我们要找的程序子句。因此，现在只需证明，存在□的一个由 G 开始从 S 出发的线性证明。对不可满足的集合 S 的文字数目作归纳，我们得到了一个比定理 10.7 更强的结论。

引理 10.11 如果 T 是一个 Horn 子句集合，G 是一个使 $T\cup\{G\}\in$ UNSAT 但 $T\in$ SAT 的目标子句，那么存在□的一个由 C 开始从 $T\cup\{G\}$ 出发的线性消解演绎。

证明 和前面一样，根据紧致性定理，假设 T 是有穷的。对 T 中文字的数目作归纳。由定理 10.7 的证明知，对某个正文字 p，T 包含了事实 $\{p\}$，同时 $T'=(T\cup\{G\})^p=T^p\cup\{G\}^p$ 是由 Horn 子句构成的一个不可满足的集合。(因为 G 是一个目标子句，它不含正文字，于是 $\{G\}^p$ 就是 $\{G-\{\bar{p}\}\}$)。由于 T 可满足并且包含 $\{p\}$，所以满足 T 的指派同样满足 T^p(根据引理 8.19 的必要性证明)。因此，我们可以应用归纳假设到 T'，从而得到□的一个由 $G-\{\bar{p}\}$ 开始从 T' 出发的线性证明。如果该证明尚不是我们要找的证明，我们可以像定理 8.22 或定理 10.7 的证明那样，将其转化成 $\{\bar{p}\}$ 的由 G 开始从 T 出发的证明。在此基础上再扩展一步，添加新的边子句 $\{p\}\in T$，与 $\{\bar{p}\}$ 消解得到□。 □

根据引理 10.5，任意程序子句集合均可满足，因此利用引理 10.11 即可证明定理 10.10。 □

现在我们知道了 PROLOG 中消解证明的一般模式：线性插入消解。在给出实现 PROLOG 的精确机制之前，还有两点需要考虑。其中最重要的一点即是，PROLOG 并非限制在命题逻辑上，它也用到了谓词和变元。这将在下一章中讨论。另外一点更加专业，它涉及排序，又分为两类。第一类是关于消解执行过程中子句的实际表示以及所要消解文字的选择。第二类是关于寻找线性证明时的序：搜索和回溯。

首先考虑子句的表示。消解的抽象表示是把子句看作文字的集合。作为集合，子句在本质上是无序的。然而机器通常把子句存储为文字的序列。此外，它还把子句作为序列而非集合来处理。因此，诸如并运算这样的集合操作势必要被有关序列的某种归并程序所替代。特别地，当 $G=\{\neg A_0, \neg A_1, \cdots, \neg A_n\}$ 与 $H=\{B, \neg B_0, \cdots, \neg B_m\}$(视作有序子句)进行消解时，比如说 $A_i=\neg B$，编译器只是把 A_i 替换成 $\neg B_0, \cdots, \neg B_m$。于是其消解式(作为有序子句)即为 $\{\neg A_0, \neg A_1, \cdots, \neg A_{i-1}, \neg B_0, \cdots, \neg B_m, \neg A_{i+1}, \cdots, \neg A_n\}$。除序自身之外，我们应当注意，从该角度出发处理子句会导致重复出现。例如，B_j 中的一个文字与某个 $A_k(k\neq i)$ 相同。PROLOG 的执行过程并不检查此类重复；它仅把文字的所有出现放到有序子句的适当位置。(有序子句有时是指确定子句(definite clause)，因此在下面一个定义中，用 LD 表示线性确定。)这种子句排序并未引起本质变化。我们将在下面的定义和引理中讨论它。

仍然用 T 表示 Horn 子句的集合，P 表示程序子句的集合，而 G 表示目标子句。

定义 10.12 *如果给定有序子句的集合 $P\cup\{G\}$，那么 $P\cup\{G\}$ 的* **LD-消解反驳**(LD-resolution refutation)*是指满足下列条件的有序子句 G_i 和 C_i 组成的序列 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$：$G_0=G$，$G_{n+1}=\square$，并且*

(i)每个 G_i，$i\leq n$ 是一个长度为 $n(i)+1$ 的有序目标子句 $\{\neg A_{i,0}, \cdots, \neg A_{i,n(i)}\}$。

(ii)每个 $C_i=\{B_i, \neg B_{i,0}, \cdots, \neg B_{i,m(i)}\}$ 是从 P 出发的一个长度为 $m(i)+2$ 的有序程序子句。(我们考虑到了 $C_i=\{B_i\}$ 的可能性，即 $m(i)=-1$。)

(iii)对每个 $i<n$，存在有序子句 G_i 和 C_i 的一个消解，消解式为有序子句 G_{i+1}(长度为 $n(i)+m(i)+1$)，记作 $\{\neg A_{i,0}, \cdots, \neg A_{i,k-1}, \neg B_{i,0}, \cdots, \neg B_{i,m(i)}, \neg A_{i,k+1}, \cdots, \neg A_{i,n(i)}\}$。(在这一步，我们对 $B_i=A_{i,k}$ 进行消解。)

引理 10.13 *如果 $P\cup\{G\}\in$ UNSAT，那么存在 $P\cup\{G\}$ 的一个由 G 开始的 LD-消解反驳。*

证明 留作习题 1。对 $P\cup\{G\}$ 的 LI-消解反驳的长度作归纳。(注意，在消解的每一步仅能消解一个程序子句与一个目标子句。每个中间子句必然是目标子句，而每个边子句必然是程序子句。) □

我们下一个任务是描述如何在一个 LD-消解证明里选取 G_i 中要消解的文字。PROLOG 的所有执行程序本质上所用的选择规则，都是始终对有序目标子句 G 中的第一个文字(在我们的标记下，它是指 G_i 中最左端的文字)进行消解。C_i 与 G_i 的消解式中的文字排序如定义 10.12所述。我们把有序子句的这种线性插入消解叫做 SLD-消解(SLD-resolution)。(S 代表选择(selection)。)更一般地，我们可以考虑任意的选择规则，即从每个有序目标子句中选择一个文字的任意函数。

定义 10.14 *$P\cup\{G\}$ 的基于(选择规则)R 的* **SLD-消解反驳**(SLD-resolution refutation)*是指满足 $G_0=G$，$G_{n+1}=\square$ 的 LD-消解证明 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$，并且 $R(G_i)$ 是该证明在第 $(i+1)$ 步要消解的文字。(如果没有提到 R，我们默认采用标准选择规则，即始终选择最左端的文字。)*

定理 10.15(PROLOG 中 SLD -反驳的完全性)　如果 $P\cup\{G\}\in \mathrm{UNSAT}$，$R$ 是任意的选择规则，那么存在 $P\cup\{G\}$ 的一个基于 R 的 SLD -消解反驳。

证明　由引理 10.13 知，存在 $P\cup\{G\}$ 的一个由 G 开始的 LD -消解反驳。我们对这般证明的长度作归纳(对任意 P 和 G)，证明存在基于 R 的 SLD -消解反驳。对 $n=1$，无须证明。因为 $G=G_0$ 是单元子句，所有规则 R 都从 G_0 做同样的选择。令 $\langle G_0, C_0\rangle$，…，$\langle G_n, C_n\rangle$ 为 $\{G_0\}\cup P$ 的一个长度为 n 的 LD -消解反驳，这里子句的记法如定义 10.12 所示。假设选择规则 R 从 G_0 中选择文字 $\neg A_{0,k}$。由于 $G_{n+1}=\square$，所以一定存在某个 $j<n$，在此时消解 $\neg A_{0,k}$。如果 $j=0$，根据归纳，已经证明。假设 $j\geqslant 1$，考虑对 G_0 与 $C_j=\{B_j, \neg B_{j,0}, \cdots, \neg B_{j,m(j)}\}$ 消解 $B_j=A_{0,k}$，得到如下消解式 C：

$$C=\{\neg A_{0,0},\cdots,\neg A_{0,k-1},\neg B_{j,0},\cdots,\neg B_{j,m(j)},\neg A_{0,k+1},\cdots,\neg A_{0,n(0)}\}$$

我们断言：存在一个由 C 开始从 $P\cup\{C\}$ 出发的长度为 $n-1$ 的 LD -消解反驳。欲证明该断言，我们只需以 C_0，…，C_{j-1} 为边子句，依次消解之。每次所消解的文字与由 G 开始的原证明中每次消解的文字对应相同。唯一的变化在于，在该消解的中间子句里，用子句序列 $\neg B_{j,0},\cdots$，$\neg B_{j,m(j)}$ 替换 $\neg A_{0,k}$。依次消解完边子句 C_0，…，C_{j-1} 之后，我们得到的 G_{j+1} 刚好与原证明中消解完 C_j 得到的结果相同。于是可以继续原证明，依次消解边子句 C_{j+1}，…，C_n。这个过程构成了一个由 C 开始长度为 $n-1$ 的 LD -消解反驳。根据归纳，它可以被某个基于 R 的 SLD -消解替代。添加该 SLD -消解到前面所述的 G_0 与 C_j 的单步消解之后，我们就得到了由 $G=G_0$ 开始基于 R 的从 $P\cup\{G\}$ 出发的 SLD -消解。　□

现在我们知道，当一个问题"? - A_1，…，A_n."输进去，PROLOG 编译器是如何工作的。它搜索□的一个由 G 开始从当前程序 P 与目标子句 $G=\{\neg A_1, \cdots, \neg A_n\}$ 出发的 SLD -消解证明。关于 PROLOG 编译器的工作过程，我们唯一不清楚的就是，它如何组织搜索。在 SLD -消解的第 i 步，唯一需要决定的是，为了消掉当前目标子句 G_i 最左端的文字，我们选择 P 中哪个子句来跟 G_i 进行消解。因此，我们可以将所有可能的 SLD -推导空间画成一棵标签树 T。T 的根标识成 G。如果 T 的任意一个节点标识成 G'，那么其直接后继分别由 G' 与 P 中各种可能的子句选择在消解 G' 最左端文字时得到的那些消解式所标识。我们把这样的树称作 P 和 G 的 SLD -树。

例 10.16(SLD -树)　作为一个简单例子，考虑程序 P_0：

```
p :- q,r.   (1)
p :- s.     (2)
q.          (3)
q :- s.     (4)
r.          (5)
s :- t.     (6)
s.          (7)
```

假设将 $G=\{\neg p\}$ 作为目标子句，图 22 给出了相对应的 SLD -树。沿着每个分支我们指出消解所用到的 P_0 中的子句。这里我们约定，节点的后继从左往右的排列顺序与生成各后继所用到的程序子句在 P_0 中的出现顺序相一致。*成功路径*(success path)是指那些以□为终结的路径。*失败路径*(failure path)是指该路径以子句 G' 为终结且 P 中没有子句能够消掉 G' 最左端的项。在本例中总共有 5 条可能路径，2 条失败，3 条成功。

PROLOG定理证明器在搜索SLD-树的成功路径时，总是最先尝试最左边的路径。换句话说，它试图利用P中第一个可能子句来和当前的G进行消解。如图22所示，沿着路径(1)，(3)，(5)可以得到肯定的答案“yes”。如果定理证明器遇到一个失败点(即非□且无消解可以进行)，它就回溯，即沿原路返回至节点N，N有尚未尝试消解的直接后继。然后证明器沿N的最左端的尚未尝试消解的直接后继继续运行。反复该过程，直至找到一条成功路径。

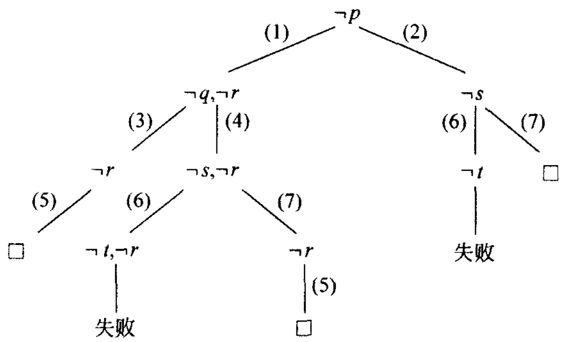


图 22

例10.17(回溯) 如果从上述程序P_0中略去子句(3)，记新程序为P_1，那么就可以得到一个新的SLD-树，如图23所示。

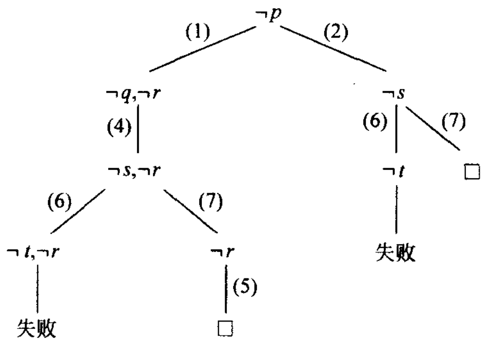


图 23

在此情形下，定理证明器首先尝试路径(1)，(4)，(6)，失败。然后回溯到¬s，¬r，尝试路径(7)，(5)，成功，给出肯定答案“yes”。

假设PROLOG编译器已经启动搜索，当它找到一个答案时，我们就在计算机光标处输入分号“;”。然后编译器会继续回溯，在树中尚未经历搜索的部分寻找另一个消解反驳。答案“no”意味着已经不存在其他证明，而答案“yes”则意味着已经找到了另一个证明，此时我们可以再次输入“;”使得编译器继续搜索新的证明。在上例中，找到沿路径(1)，(4)，(7)，(5)的证明之后，定理编译器回答“yes”。如果我们输入分号搜索另一个证明，它会沿原路回溯至顶端节点，然后尝试路径(2)。接下来它沿路径(6)，失败，回溯至¬s，再沿路径(7)，成功，从而得到另一个肯定答案“yes”。如果再次输入分号，则最终得到答案“no”。

如果PROLOG搜索整棵SLD-树而没有找到一条导出□的路径，那么它在我们第一次询问时就给出答案“no”。由一般消解的完全性定理知，在此情形下$P\cup\{G\}$是可满足的，因而(根据定理10.6)我们询问的问题不是P的逻辑后承。

这一类型的搜索过程叫做深度优先(depth-first)搜索，因为它总是试图沿着树中一条路径尽可能深地走到末端，然后才沿其他分支继续搜索。与之相对地，如果按照$\neg p$；$\neg q$，$\neg r$;$\neg s$；$\neg r$；$\neg s$，$\neg r$；$\neg t$；□；□；$\neg t$，$\neg r$；$\neg r$；失败；失败；□这样的顺序对如图22所示的树进行搜索，则该搜索叫做广度优先(breadth-first)搜索。显然，很多混合策略也是可能的。在我们的例子中，深度优先要比广度优先快得多(3步对6步)。事实上，这是一个普遍现象，深度优先通常都比广度优先快很多。所以执行程序选择用深度优先搜索。然而这种策略的时间花费也可能非常高。在广度优先搜索中，对有穷程序P，如果存在一条路径是以□结尾，很明显我们最终一定能够找到它。与之相对地，深度优先搜索程序却不是完全的：可能存在一条路径以□结尾，我们的搜索一直进行却找不到□。

例 10.18(深度优先搜索失败) 考虑下面的简单程序：

```
q :- r.   (1)
r :- q.   (2)
q.        (3)
```

把深度优先的搜索程序应用到开始子句$\neg q$，该程序将会在$\neg q$和$\neg r$之间循环，它永远找不到子句(3)所提供的矛盾。

这个例子看起来比较容易确定子句的排序，并且它也正依赖于程序中子句的排序。不幸的是，重排子句并非总能生成一个终结程序，即便确实存在着一个正确的证明。(第三章第二节的程序P_5就是一个例子。)但是我们还不能完全体会这些问题带来的影响，除非我们着手处理完全的PROLOG，而不仅仅局限于命题逻辑。事实上，也只有介绍了谓词和变元，才会看到PROLOG的真正威力。现在我们开始考虑这些事情，首先研究完全谓词逻辑的一般情形，然后专门考察PROLOG。

习题

1. 证明引理10.13。
2. 在自然语言(汉语)的环境下考虑下面的语句：如果国会拒绝出台新法令，那么罢工就不会结束，除非它的持续时间超过一年并且公司总裁辞职。
   (a)用以下三种方式表达该语句：
      (i)命题演算
      (ii)合取范式
      (iii)PROLOG子句。
   (b)假设国会已经拒绝出台新法令，而罢工尚未超过一个月。把以上条件添加到你的列表(在(i)，(ii)，(iii)中)，然后利用表演绎来检查罢工是否已经结束。(可以尝试从一些适当的前件出发构造一个演绎，或者给出某个适当合取的一般形式，但要注意问题的形式化表述。)
   (c)假设去掉“国会拒绝新法令出台”这个条件，取而代之的是：罢工已经结束，但是公司总裁没有辞职。利用表演绎来检查罢工的持续时间是否超过了一年。
   (d)在(b)和(c)中，如果PROLOG数据库里已有一列相关的子句，那么为了得到所求答案，你会如何询问？
3. 专家系统的一个成功应用就是分析哪些化学合成是可能的。这里我们考虑一个非常简单的例子。
   我们知道下列化学反应可以进行：
   (1)$MgO + H_2 \rightarrow Mg + H_2O$
   (2)$C + O_2 \rightarrow CO_2$
   (3)$CO_2 + H_2O \rightarrow H_2CO_3$

(a)假设现有一些 MgO, H_2, O_2 和 C, 请用命题逻辑公式表达以上规则和假设。在所写公式中, 我们把断言(比如说我们有某种化学药品)和蕴涵理解为: 如果有了某个假设, 我们就能得到其结论。(因此, (1)即为 $MgO \wedge H_2 \rightarrow Mg \wedge H_2O$。)

(b)把以上化学反应式写成子句形式, 然后组合成一个 PROLOG 程序。

(c)给出一个(树形或线性的)消解证明, 说明我们可以得到 H_2CO_3。

4. 将下面的信息表述成一个 PROLOG 程序, 使得如果告诉你 Jones 已经病危并且 Smith 已经离开其身边, 你能够判定董事会是否会宣布解散。

如果 Jones 生病或者 Smith 外出, 那么一旦 Robinson 醒来并接管事务, 董事会就宣布解散。如果 Patterson 来了, 他会叫醒 Robinson, 但只有 Jones 生病他才会来。另一方面, Townsend 与 Smith 是不可分开的, 如果 Townsend 离开了, Robinson 将不得不接管事务。

从你的程序以及上面的条件假设出发, 给出一个消解证明, 来说明董事会将宣布解散。

5. 将下面的信息表述成一个 PROLOG 程序:

如果国会出台了条文禁令并且总统切实地实现, 那么如果没有贸易保护主义者和高利率支持者的游说活动, 预算和贸易赤字都将减少。公众的强烈抗议会使国会出台条文禁令并强迫总统切实执行。如果生产效率提高并且美元继续贬值, 贸易保护主义者就会保持安静。(提示: 可以先用命题逻辑表述, 然后再进行适当地转化。)

如何添加 PROLOG 子句来反映如下事实: 公众对赤字怨声载道, 国际市场美元继续下跌, 而生产效率正在提高?

如何向 PROLOG 程序发出如下询问: 贸易赤字是否会减少?

给出消解反驳来说明贸易赤字将会减少。

6. 对问题"? - p. "与下面的程序, 画这样一棵 SLD -树: 它利用标准选择规则(始终消解最左端的文字)尝试了所有可能的 SLD -反驳。

p: - s, t.	(1)
p: - q.	(2)
q.	(3)
q: - r.	(4)
r: - w.	(5)
r.	(6)
s.	(7)
t: - w.	(8)

如果每次得到"yes"的回答我们都输入一个分号, 那么 PROLOG 编译器会按照何种顺序搜索这棵树? 它又会给出怎样的输出?

进一步阅读建议

关于我们在处理序和树时所需的一般集合论背景, 参见本书第六章。若要获得更多关于序、偏序和树的知识, 可以参考 Birkhoff[1973, 3.8], 也可以查阅参考文献[5.2]所列的任何一本关于逻辑与计算机科学的教材。

关于早期的命题逻辑, 参阅 Boole[1952, 2.3]以及 Post[1921, 2.3]。

关于各种相互可替换的逻辑形式, 参阅以下文献的命题逻辑部分:

表: Beth[1962, 3.2]、Smullyan[1968, 3.2]以及 Fitting[1987, 4.2]。

公理与推理规则: Hilbert 和 Ackermann[1950, 3.2]、Mendelson[1964, 3.2]以及 Enderton[1972, 3.2]。更进一步, 可以参阅 Kleene[1971, 3.2]、Monk[1976, 3.2]或者 Shoenfield[1967, 3.2]。

消解：Chang 和 Lee[1973，5.7]、J. A. Robinson[1979，5.2]、Lewis 和 Papadimitriou[1981，5.2]以及 Maier 和 Warren[1988，5.4]。

自然演绎：Prawitz[1965，3.5]，更进一步，可以参阅 Girard[1987，3.5]和 Girard 等[1989，3.5]。

序列：Gallier[1986，5.2]、Manaster[1975，3.2]、Girard[1987，3.5]以及 Girard 等[1989，3.5]。

关于消解与 Horn 逻辑的有关问题，参阅 Kowalski[1979，5.4]。

关于布尔代数及其与命题逻辑的联系，参阅 Halmos[1974，3.8]、Sikorski[1969，3.8]或者 Rasiowa 和 Sikorski[1963，3.8]。

近来关于命题逻辑的研究重新成为热点，这主要是受到了不同证明系统的证明复杂性研究的带动。Urquhart[1995，3.5]是这方面工作的一个很好的综述。

第二章 谓词逻辑

第一节 谓词和量词

谓词逻辑涵盖了很多数学推理。我们已经对对象所具有的性质以及对象间的关系有了直观概念。任何这样的性质或者关系都是**谓词**(predicate)的例子。性质和关系的区别仅在于谓词中元的个数。一元谓词表示对象的性质，二元谓词表示对象二元组之间的关系，一般地，n-元谓词表示对象的 n-元组之间的关系。0-元谓词表示什么？对该问题的一个合理答案是：它们是命题。这里重点在于，它们是不依赖于任何变元的事实的表述。因此，例如，若讨论自然数，则可以令 $\varphi(x, y)$ 表示二元关系(谓词)"x 小于 y"。在此情形下，$\varphi(3, y)$ 表示 y 的性质(一元谓词)"3 小于 y"，而 $\varphi(3, 4)$ 表示(真)命题(0-元谓词)"3 小于 4"。

在此讨论中 x 和 y 被当作**变元**(variable)，而 3 和 4 则被当作**常元**(constant)。变元很像自然语言中的名词符号。在下面的约定下，变元的功能更易理解：对任何一个特定的推理，首先指定对象的一个非空定义域。出现在上述讨论中的所有变元的取值遍及该定义域。常元表示该定义域里对象(个体)的名字。在此约定下，n-元谓词可以看作由论域上的 n-元组构成的集合，并对这些集合保持成立。因此，例如，一元谓词可以看作使得该性质为真的那些元素所组成的定义域的一个子集。0-元谓词是指论域上的那些事实，它在谓词逻辑中叫做**语句**(sentence)而非命题。

数学推理中另外一个重要的对象集合是**函数**(function)集合。例如，$f(1, 2)$ 可以代表 1 与 2 的和。函数也有元数，它对应着该函数输入中的变元个数。自然数的一般加法是二元函数，乘法亦然。但是自然数的减法甚至不是函数，问题在于两个自然数的差可能不是自然数。我们要求所考虑的函数的输出始终是论域中的元素(不过，可以根据需要任意改变论域)。另一方面，并非定义域中的每个元素都要成为函数的一个取值。再看一个例子，考虑三元函数 $g(x, y, z) = x \cdot y + z$。这里，变元在函数中与其在谓词中的作用类似。和上面处理二元关系 φ 类似，从 g 出发，用常元替换一些变元，从而定义二元函数和一元函数如下：$g(1, y, 1)$ 是一元函数 $y+1$，$g(x, y, 0)$ 是乘法。那么如何看待 $g(1, 1, 0)$？很明显它是常元 1。因此，正如把命题看作 0-元谓词一样，可以把常元看作 0-元函数。常元不依赖于任何输入，仅表示论域中的元素。更一般地，把所有由函数符号、常元以及变元生成的符号，例如 $f(x, g(y, y))$，叫做**项**(term)。我们认为项取遍论域(或者可能只是论域的某个子集，即通常所说的函数值域)。

和命题的情形一样，可以用真值函数联结词从简单谓词出发构造复合谓词。例如，$\varphi(x, y)$ 仍表示关系"x 小于 y"，而 $\psi(x, y)$ 表示关系"x 整除 y"，那么 $(\varphi(x, y) \wedge \psi(x, y))$ 就是一个新的二元谓词，并且含义明确。除了真值函数联结词外，谓词逻辑还用到了另外两个谓词构造符：

(i) **全称量词**(universal quantifier)"$\forall$"，是指"对所有"。

(ii) **存在量词**(existential quantifier)"$\exists$"，是指"存在"。

例 1.1

(i)令论域为自然数集$\mathcal{N}$，令“$\varphi(x, y)$”表示“$x<y$”，$f(x, y)$表示二元函数$x+y$，而a，b，c分别为表示0，1，2的常元：

(a)$((\exists x)\varphi(x, y))$是$y$的一元谓词，即存在一个自然数小于$y$。它等价于“$y$不为0”。语句$((\forall x)((\exists y)\varphi(x, y)))$为真(0-元谓词)，即对任何自然数$x$，存在自然数$y$大于$x$。

(b)语句$((\forall x)\varphi(x, f(x, b)))$是指对每一个$x$，$x<x+1$，即每个自然数都小于其后继。$\varphi(y, f(y, y))$也是$y$的一元谓词，即$y<y+y$。这个谓词也等价于“$y$不为0”。

(ii)令论域为有理数集$\mathcal{Q}$。仍用$\varphi(x, y)$表示$x<y$，$f(x, y)$表示加法$(x+y)$，$g(x, y)$表示除法$(x\div y)$，而a，b，c分别为表示0，1，2的常元。

(a)三元谓词$(\varphi(x, y)\wedge\varphi(y, z))$是指$x<y$且$y<z$。

(b)二元谓词$((\exists y)(\varphi(x, y)\wedge\varphi(y, z)))$是指$x$与$z$之间存在一个有理数。一元谓词$((\forall x)(\varphi(x, z)\rightarrow((\exists y)(\varphi(x, y)\wedge\varphi(y, z)))$表示$z$的性质，即对任意$x$，如果$x$小于$z$，那么它们之间存在一个有理数。

(c)语句$((\forall x)((\forall y)(\varphi(x, y)\rightarrow(\varphi(x, g(f(x, y), c))\wedge\varphi(g(f(x, y), c), y)))))$是指对每个$x$，$y$，如果$x<y$，那么$x<\frac{x+y}{2}<y$。

(d)一元谓词$\varphi(y, f(y, y))$是指$y<y+y$。但是要注意，在此论域内该谓词等价于y为正。

第二节　语言：项和公式

本节首先给出一种适合于谓词逻辑语言的形式定义，然后选择那些我们认为有意义的“好”的字符串，归纳定义出谓词逻辑公式。

定义 2.1　**语言**(language)$\mathcal{L}$是由下列各种基本符号组成的：

(i)变元：x，y，z，v，x_0，x_1，…，y_0，y_1，…，…(无穷集合)。

(ii)常元：c，d，c_0，d_0，…(由其中成员构成的任意集合)。

(iii)联结词：$\wedge$，$\neg$，$\vee$，$\rightarrow$，$\leftrightarrow$。

(iv)量词：$\forall$，$\exists$。

(v)谓词符号：P，Q，R，P_1，P_2，…(对每个元数$n=1, 2, \cdots$，由其相应的成员构成的某个集合。语言中至少要存在一个谓词符号，但对每个元数，谓词符号的数目没有限制)。

(vi)函数符号：f，g，h，f_0，f_1，…，g_0，…(对每个元数$n=1, 2, \cdots$，由其相应的成员构成的任意集合。0-元函数符号就是(ii)中约定的常元。常元符号的集合可能为空、有穷或无穷)。

(vii)标点符号：逗号“,”以及左右括号“(”和“)”。

注意，我们不再有命题字母(应看作0-元谓词)，它们在谓词逻辑中已不必要。真(假)命题可以被始终为真(假)的语句所替代，例如$\alpha\vee\neg\alpha(\alpha\wedge\neg\alpha)$。(定理4.8指出，命题逻辑可以嵌套至谓词逻辑。)

在定义语言$\mathcal{L}$中的公式之前，先定义$\mathcal{L}$中的项，即用来表示论域中元素的那些符号。本节采用归纳定义。(喜欢运用形成树方法的读者可以跳过这里的传统语法定义，直接到下一

节，将那里的表述当作定义，并忽略两种定义之间的等价性证明。)

定义 2.2　项(term)

(i)所有变元都是项。

(ii)所有常元符号都是项。

(iii)如果f是n-元函数符号($n=1, 2, \cdots$)且$t_1, \cdots, t_n$是项，那么$f(t_1, \cdots, t_n)$也是项。

定义 2.3　不含变元的项叫做**无变元项**(variable-free term)或**基本项**(ground term)。

基本项是论域中特殊命名的元素，是常元以及通过对常元应用定义 2.2(iii)中的函数符号而得到的那些项。

公式定义的基本情形如下：

定义 2.4　原子公式(atomic formula)是指形如$R(t_1, \cdots, t_n)$的表达式，这里R是n-元谓词符号，$t_1, \cdots, t_n$是项。

现在给出公式完整的归纳定义。

定义 2.5　公式(formula)

(i)每个原子公式都是公式。

(ii)如果α，β是公式，那么$(\alpha \wedge \beta)$，$(\alpha \rightarrow \beta)$，$(\alpha \leftrightarrow \beta)$，$(\neg \alpha)$和$(\alpha \vee \beta)$也都是公式。

(iii)如果v是变元，α是公式，那么$((\exists v)\alpha)$和$((\forall v)\alpha)$也是公式。

定义 2.6

(i)公式φ的**子公式**(subformula)是φ中的一段连续符号序列，且其自身是公式。

(ii)变元v的某个**出现**在公式φ中是**约束的**(bound)，如果存在φ的一个子公式ψ，它包含v的这次出现并以$((\forall v)$或$((\exists v)$为开端。(根据这个定义，$\forall v$或$\exists v$中的v也是约束的。)v的某个出现在公式φ中是**自由的**(free)，如果它不是约束的。

(iii)变元v在φ中**自由出现**(occur free)，如果它在φ中至少有一个出现是自由的。

(iv)谓词逻辑中的**语句**是指任何变元在其中均无自由出现的公式，即每一个变元的所有出现在其中都是约束的。

(v)**开公式**(open formula)是指不带量词的公式。

语句的定义包含了如下思想：每个语句对应一个具有唯一固定含义和真值的公式。注意，要生成元数大于 0 的谓词，唯一的方法就是使用变元。变元总是伴随着量词出现，具体来说，它们仅出现在下面两种表述中：“存在一个x，使得……”或者“对所有x，……为真”。下面定义的主要思想是用一些项来替换变元，从而生成可能具有更小元数的谓词(与第一节中的做法相同)。

定义 2.7 替换(substitution，或**实例化**，instantiation)　如果φ是公式，v是变元，用$\varphi(v)$来表示v在φ中自由出现这一事实。如果t是项，则$\varphi(t)$(更准确地，$\varphi(v/t)$)是将v在φ中的所有自由出现替换成t所得到的结果。我们把$\varphi(t)$叫做φ的**实例**(instance)。如果$\varphi(t)$不包含自由变元，那么称之为φ的**基本实例**(ground instance)。

在做替换时，有一个地方必须注意。

定义 2.8　假设项t包含某个变元x(必须在t中自由)的一个出现。我们说对$\varphi(v)$中的自由变元v，t是**可替换的**(substitutable)，如果t中x的所有出现都在$\varphi(v/t)$中保持自由。

注意，基本项对任何自由变元都是可替换的。如果(带变元的)项t在φ中是不可替换

的，那么通过定义公式语义，会更加清楚地看到用 t 进行替换所引发的问题。我们来看两个例子。

例 2.9

(i) 首先考虑一元谓词 $\psi(y)=((\exists x)\varphi(x, y))$，这里的记号表示与例 1.1(i) 相同。毫无疑问，用 z 或 2 甚至 $f(w, w)$ 来替换 y，分别得到 $((\exists x)\varphi(x, z))$，$((\exists x)\varphi(x, 2))$ 和 $((\exists x)\varphi(x, f(w, w)))$。这些公式是说 z，2 和 $w+w$ 不为 0，正如我们的期望。然而，如果用 $f(x, x)$ 来替换 y，那么将会得到 $((\exists x)\varphi(x, f(x, x)))$。该公式关于 x 和 $x+x$ 什么也没说，它仅是一个真语句，断言存在某个 x，使得 $x<x+x$。

(ii) 下面考虑带有常元 0 和 1 的整数集 $\mathcal{Z}$ 上的语言。一元函数符号 s 表示后继，谓词 $A(x, y, z)$ 意指 $x+y=z$。令 φ 是在 $\mathcal{Z}$ 中为真的语句 $\forall x\exists yA(x, y, 0)$。作为一个真的全称语句，$\varphi$ 应该对任何对象为真。事实上，任何符合定义的替换都会生成一个在 $\mathcal{Z}$ 中永真的公式。另一方面，如果违反可替换性的定义，用 $s(y)$ 来替换 x，那么就会得到 $\forall x\exists yA(s(y), y, 0)$，它在 $\mathcal{Z}$ 中为假。

例 2.10

(i) y 在公式 $((\forall x)R(x, y))$ 中自由出现，而 x 则不然。公式 $((\exists y)((\forall x)R(x, y)))$ 没有自由变元，它是一个语句。

(ii) 一个变元在一个公式中可以同时有自由出现和约束出现，例如，公式 $(((\forall x)R(x, y))\vee((\exists y)R(x, y)))$ 中的 x 和 y。

(iii) 如果 $\varphi(x)$ 是 $(((\exists y)R(x, y))\wedge((\forall z)\neg Q(x, z)))$，$t=f(w, u)$，那么 $\varphi(t)=\varphi(x/t)$ 是 $(((\exists y)R(f(w, u), y))\wedge((\forall z)\neg Q(f(w, u), z)))$。但是对 $\varphi(x)$ 中的 x，项 $g(y, s(y))$ 不是可替换的。

在本节习题之后，做替换时通常省去公式中的括号以增强可读性。例如，把 $((\exists x)\varphi(x))$ 写成 $\exists x\varphi(x)$。不过，在进一步研究之前，我们先来证明项或公式的读取无歧义。

命题 2.11 *如果项 s 是项 t 的初始段，即 $s\subseteq t$，那么 $s=t$。*

证明 如果 s 是变元或常元符号，那么此命题是显然的。否则，s 必是 $f(s_1, \cdots, s_n)$ 的形式，因此 s 的长度至少为2。现在，如果 $s\neq t$，那么 s 是 t 的真初始段，即 $s\subset t$，而这与习题 6 所证明的括号性质相矛盾。 □

定理 2.12(项的唯一可读性) *每个项 s，或者是变元符号，或者是常元符号，或者形如 $f(s_1, \cdots, s_n)$，这里 f，n 以及 $s_i(1\leqslant i\leqslant n)$ 都是唯一确定的。*

证明 如果 s 既非变元也非常元符号，那么根据项的定义，它形如 $f(s_1, \cdots, s_n)$。如果它同时具有形式 $g(t_1, \cdots, t_m)$，那么显然 $f=g$，从而 $n=m$。更进一步，显然有 $s_1\subseteq t_1$ 或者 $t_1\subseteq s_1$。根据命题 2.11，两种情形下均有 $s=t$。现在可以采用同样的办法依次得到 $s_i=t_i$ $(i\leqslant n)$，从而唯一性得证。 □

命题 2.13 *如果公式 α 是公式 γ 的初始段，即 $\alpha\subseteq\gamma$，那么 $\alpha=\gamma$。*

证明 显然，每个公式的长度都至少为 2，因此，如果 α 是 γ 的真初始段，那么它就与习题 9 所证明的括号性质相矛盾。 □

定理 2.14(公式的唯一可读性) *每个公式 ψ 具有下列形式之一：原子公式(即形如 $R(t_1, \cdots, t_n)$，R 为 n-元谓词符号，$t_1, \cdots, t_n$ 为项)，$(\alpha\wedge\beta)$，$(\alpha\rightarrow\beta)$，$(\alpha\leftrightarrow\beta)$，$(\neg\alpha)$，*

$(\alpha \vee \beta)$，$((\exists v)\alpha)$或者$((\forall v)\alpha)$（这里α，β是公式，v是变元）。此外，ψ的这些形式的相关“分支”（即原子公式ψ中的R，n和$t_i(1 \leq i \leq n)$以及ψ的其他可能形式中的公式α，β和变元v）是唯一确定的。

证明 根据公式的定义易知，ψ必是这些形式之一。如果它是原子公式，那么它不可能同时具有其他形式，因为其他所有形式都是以左括号开始，而原子公式则不然。为了说明唯一性，假设ψ同时具有形式$R(t_1, \cdots, t_n)$和$P(s_1, \cdots, s_m)$，显然，$R=P$，从而$n=m$。因此有$t_1 \subseteq s_1$或者$s_1 \subseteq t_1$，根据命题2.11，可以得到$t_1=s_1$。同理可以证明，对每个$i \leq n$都有$t_i=s_i$，唯一性得证。如果ψ不是原子公式，那么它必是其余形式之一。例如，假设ψ具有形式$(\alpha \wedge \beta)$。如果它同时具有形式$(\gamma \to \delta)$，$(\gamma \leftrightarrow \delta)$或$(\gamma \vee \delta)$，那么$\alpha \subset \gamma$或$\gamma \subset \alpha$，与命题2.13相矛盾。如果$\psi$同时具有形式$(\neg \gamma)$，$((\exists v)\gamma)$或$((\forall v)\gamma)$，则与习题7相矛盾。为了说明$\psi$的“分支”的唯一性，假设$\psi$同时具有形式$(\gamma \wedge \delta)$。显然，$\alpha \subseteq \gamma$或者$\gamma \subseteq \alpha$，因此，由命题2.13可知$\alpha=\gamma$。同理可以推出$\beta=\delta$。

关于ψ的其他形式，推理过程与此相似，留作习题10。 □

习题

对于习题1～5，按照定义2.1指定所用语言。

1. 下面哪些是项？
   a) x
   b) xy
   c) c
   d) $P(c)$
   e) $f(x, d)$
   f) $(\forall x)(R(c))$
   g) $g(c, f(y, z))$
   h) $g(R, d)$
2. 下面哪些公式是按照定义2.5完整写出来的？
   a) $f(x, c)$
   b) $R(c, f(d, z))$
   c) $(\exists y)(P(c))$
   d) $\forall x(P(x))$
   e) $(\neg R(z, f(w))$
   f) $((\exists x)(((\forall y)P(z))\to R(x, y)))$
3. 对习题2中的每个公式，写出其所有子公式。
4. 对习题3中的每一个子公式，哪些变元的出现是自由的？哪些是约束的？
5. 根据可替换的定义，下面哪些替换是允许的？
   a) $x/f(z, y)$，$((\exists y)(P(y) \wedge R(x, z)))$
   b) $x/g(f(z, y), a)$，$(((\exists x)(P(x) \wedge R(x, y))) \to P(x))$
   c) $x/g(f(z, y), a)$，$((\exists x)(P(x) \wedge R(x, y)))$
   d) $x/g(a, b)$，$((\exists y)(R(a, x) \wedge P(y)))$
6. 证明：每个项中的左右括号数目相等。更进一步，对项t的每个真初始段，其左括号至少和右括号一样多，并且如果初始段的长度不小于2，那么它的左括号多于右括号。
7. 证明：不存在以¬，∃，∀，(∃或(∀开始的公式。
8. 证明：每个公式的左右括号数都相等。

9. 证明：对一个公式的每个真初始段，其左括号至少和右括号一样多，并且如果初始段的长度不小于 2，那么它的左括号多于右括号。

10. 验证定理 2.14 的其余情形。

第三节　形成树、结构和列表

和在命题定义时一样，形成树表示法使得公式的形成规则更明晰，并且使得诸如“出现”这样的术语更精确。大多数 PROLOG 程序设计教材都喜欢采用该表示法。我们仍然从项的定义开始。

定义 3.1

(i) **项形成树**(term formation tree) 是有序且有穷分叉的标签树 T，树上的标签满足以下条件：

(1) T 的叶子标签是变元或常元符号。

(2) T 的每个非叶节点的标签是形如 $f(t_1, \cdots, t_n)$ 的项。

(3) 若 T 的节点标签为形如 $f(t_1, \cdots, t_n)$ 的项，则该节点在树中恰好有 n 个直接后继。它们的标签(按字典序排)依次为 $t_1, \cdots, t_n$。

(ii) 一棵项形成树**对应**(associated with) 一个项，该项标识了形成树的根节点。

例 3.2　分别与 $f(c, g(x, y))$，$h(f(d, z), g(c, a), w)$ 相对应的项形成树，如图 24 所示。

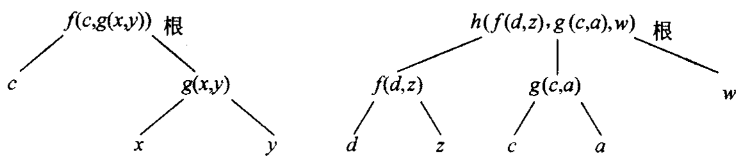


图　24

命题 3.3　每个项 t 有唯一一个与之对应的形成树。

证明　和本节其他结论的证明一样，本命题的证明是归纳法的一个简单习题，如第一章定理 2.4。我们把本节的所有证明都留作习题。本命题的证明留作习题 4。 □

命题 3.4　基本项是指那些对应形成树的所有叶节点都不含变元的项。

证明　留作习题 5。 □

原子公式处理如下：

定义 3.5

(i) **原子公式辅助形成树**(atomic formula auxiliary formation tree) 是有序且有穷分叉的深度为 1 的标签树 T，它的根节点被一个原子公式所标识。如果这样一棵树的根节点被一个 n-元关系 $R(t_1, \cdots, t_n)$ 所标识，那么它有 n 个直接后继，依次被标识为 $t_1, \cdots, t_n$。

(ii) **原子公式形成树**(atomic formula formation tree) 是通过如下过程得到的有序且有穷分叉的标签树：对辅助树上的每个标签为 t 的叶子，添加项 t 所对应的形成树除根以外的部分到该叶子。这样的一棵树对应一个原子公式，此原子公式标识了形成树的根节点。

例 3.6　图 25 给出了公式 $R(c, f(x, y), g(a, z, w))$ 所对应的形成树。

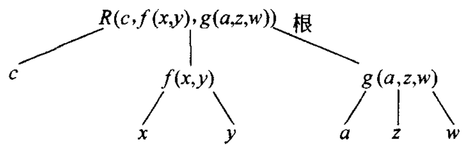


图 25

命题 3.7 每个原子公式都对应唯一的形成树。

证明 留作习题6。 □

定义 3.8

(i) **公式辅助形成树**(formula auxiliary formation tree)是有序的二叉标签树 T，且满足下列条件：

(1) T 的叶子由原子公式标识。

(2) 如果 σ 是 T 的一个非叶节点，且有一个直接后继 $\sigma \wedge 0$ 由公式 φ 所标识，那么对某个变元 v，σ 由 $\neg\varphi$，$\exists v\varphi$ 或 $\forall v\varphi$ 所标识。

(3) 如果非叶节点 σ 有两个直接后继 $\sigma \wedge 0$ 和 $\sigma \wedge 1$，分别由公式 φ 和 ψ 所标识，那么 σ 由 $\varphi \wedge \psi$，$\varphi \vee \psi$，$\varphi \rightarrow \psi$ 或 $\varphi \leftrightarrow \psi$ 所标识。

(ii) **公式形成树**(formula formation tree)是通过如下过程得到的有序标签树：对辅助树上的每个由原子公式标识的叶子，添加原子公式所对应的形成树除根以外的部分到该叶子。每棵这样的树对应一个原子公式，此原子公式标识了形成树的根节点。

(iii) **公式的深度**(depth of a formula)是指其所对应的辅助形成树的深度。

例 3.9 图26给出了公式 $\exists xR(c, f(x, y), g(a, z, w)) \wedge \forall yR(c, f(x, y), g(a, z, w))$ 所对应的形成树。

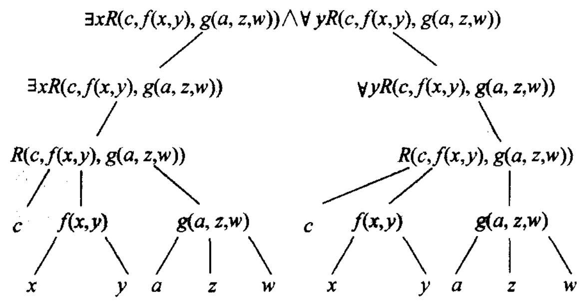


图 26

命题 3.10 每个公式都对应唯一的(辅助)形成树。

证明 留作习题7。 □

形式上，我们仍然把下面关于公式、子公式以及变元的出现等概念看作与上一节中相应的概念等价。事实上，上一节的定义可以用这里的定义替代。

命题 3.11 公式 φ 的子公式分别是与 φ 对应的辅助形成树上各节点的标签。

命题 3.12

(i) 公式 φ 中变元 v 的所有出现与 φ 的形成树上标签为 v 的叶子一一对应。(这种对应是

指 φ 中 v 的出现的印刷序，与 v 所标识的叶节点在形成树上的左右序对应。）我们也可以把某个以 v 为标签的叶子当作 v 在 φ 中的一个出现。

(ii) 变元 v 在 φ 中的一个出现是约束的，如果存在一个以 $((\forall v)$ 或 $((\exists v)$ 开始的公式 ψ，它是 φ 中 v 所标识的叶子之上的那个节点的标签。

命题 3.13 如果 φ 是公式，v 是变元，那么公式 $\varphi(v/t)$ 对应了如下形成树：用 t 对应的形成树来替换 $\varphi(v)$ 的形成树上以 v 的自由出现作标签的所有叶节点，然后对 $\varphi(v)$ 的形成树上其他节点作相应的变动。

命题 3.14 对 $\varphi(v)$ 中的 v，项 t 是可替换的，如果 t 中 x 的所有出现在 $\varphi(t)$ 中保持自由，即对 t 的形成树上以 x 的自由出现作标签的所有叶节点，在命题 3.9 所述的有 x 出现的各子公式中，x 保持自由。

以上命题的证明分别留作习题 7 ~ 11。

注意，除了在字母表中规定的函数符号和谓词符号的差别，项和原子公式对应的形成树可视为相同。两者的叶节点都由常元或变元标识，而其余节点的标签是由 n-元符号作用到其直接后继的标签得到的。PROLOG 的标准执行，乃至其他各种程序设计教材，实质上并未考虑两种符号的字母差别。项和原子公式放在一起，叫做结构（structure）。因此，可以得到一个语法上可以接受的 PROLOG 子句，例如"reading(john, reading(jack, list1))"。该子句可以用英文表述如下：John is reading Jack's first reading list。这里"reading"既作为谓词来描述是谁在阅读什么，同时又作为函数给出了阅读条目。不过在一般情况下，很难保证这样的混合使用是相容的。下一节会给出谓词逻辑的标准语义，它只在承认函数符号与谓词符号有差异的条件下有意义。由于该条件是 PROLOG 理论分析（例如，可靠性和完全性分析）的基础，并且没有证据表明采用这种混淆是必要的，因此，仍假设函数符号与谓词符号的字母表是不同的（至少在某个特殊的程序或应用中）。

例 3.15 作为典型的 PROLOG 结构的一个例子，我们简要考察配对函数（pairing function）"."，它是最重要的函数符号（或算子）之一。$.(a, b)$ 表示第一个元素为 a、第二个元素为 b 的有序对。通过重复应用配对函数，可以构造任意的列表。具体地说，算子"."在 PROLOG 中只能应用到这样的有序对上，即有序对的第二个元素已经是一个列表（有序对的第一个元素可以任取）。在整个过程的开始，PROLOG 用常元符号 [ ] 表示空列表（不含任何元素）。于是仅含元素 b 的列表就记为 $.(b, [\ ])$，从而有序对 $\langle a, b\rangle$ 被执行为 $.(a, .(b, [\ ]))$。由于这种记法烦琐，因此也可以采用下述办法来表示 PROLOG 中的列表：把列表中的元素按顺序放进方括号内，它们之间用逗号隔开。因此，$[a, b, c, d]$ 就表示由元素 a, b, c, d 按顺序排成的列表。对于配对的迭代应用（约定以空列表结尾），这种记法相当简洁。例如，把 $.(a, .(b, .(c, .(d, [\ ]))))$ 表示成 $[a, b, c, d]$。图 27 给出了它的形成树。

列表 $[a, b, c, d]$ 也可记为 $[a \mid [b, c, d]]$。竖线"|"是函数符号"."在列表结合下的另一种表示。$[X \mid Y]$ 表示这样的列表：首元素为 X，其后是列表 Y 的所有元素，按照 Y 中顺序排列。在这种记法下，新列表的首元素，即 X，叫做列表 $[X \mid Y]$ 的头；剩余元素构成的列表，即 Y，叫做尾。

如果 b 不是一个列表，那么就要避免 $[a \mid b]$ 或 $.(a, b)$ 这样的记法，因为我们通常递归地定义这种列表。递归的起始点一般是空列表 [ ]。因此，如果 b 不是一个列表，对于输入 $.(a, b)$，递归定义得到的函数永远不可能被计算到。在定义了谓词逻辑与 PROLOG 的语义之后，在第五节将继续讨论相关的例子以及对递归定义进行解释。

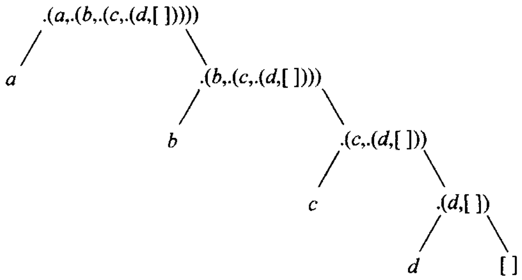


图 27

习题

1. 画出下列项对应的形成树：

a) c

b) $f(x, d)$

c) $g(f(x, d), c)$

d) $h(y, g(z, f(f(c, d), g(x, z))))$

2. 画出下列公式对应的形成树：

a) $R(c, d)$

b) $R(f(x, y), d)$

c) $R(c, d) \wedge R(f(x, y), d)$

d) $\exists y \forall z(R(x, f(c, d)) \vee \neg P(h(y)))$

e) $\forall z(R(g(x, z, z)) \rightarrow P(y)) \wedge P(z)$

3. 指出在习题 2 的树中，哪些叶子是变元的自由出现。

4. 证明命题 3.3。（提示：利用定理 2.12。）

5. 证明命题 3.4。

6. 证明命题 3.7。（提示：利用定理 2.14。）

7. 证明命题 3.10。（提示：利用定理 2.14。）

8. 证明命题 3.11。

9. 证明命题 3.12。

10. 证明命题 3.13。

11. 证明命题 3.14。

12. 证明：在谓词逻辑语言$\mathcal{L}$中，每个项 t 的长度都大于等于其所对应形成树的深度。

13. 证明：在谓词逻辑中，每个公式 φ 的长度都严格大于其所对应形成树的深度。

第四节　语义：含义与真值

谓词逻辑的语言$\mathcal{L}$由其谓词（或关系）符号及函数符号指定。一种语言可能有多种解释，不同的解释对应不同的语境或论域。因此，只含一个二元谓词 $P(x, y)$ 的语言可以具体地看作下列情形之一：

1）自然数集$\mathcal{N}$，带有 $<$。

2）有理数集$\mathcal{Q}$，带有 $<$。

3）整数集$\mathcal{Z}$，带有 $>$。

或者其他多种可能之一。例如，如果增加一个二元函数符号$f(x, y)$，那么可以把f看作是各自定义域上的$x \cdot y$，$x-y$或$\max\{x, y\}$。为了解释语言，我们必须指定一个论域，同时赋予谓词符号与函数符号一定的含义。

定义 4.1 语言$\mathcal{L}$的一个**结构**(structure)$\mathcal{A}$由四部分组成，它们分别为：一个非空定义域A；对$\mathcal{L}$中每个n-元谓词符号R，A中n-元组$(a_1, \cdots, a_n)$上的实际谓词(即关系)$R^{\mathcal{A}}$的一个指派；对$\mathcal{L}$中每个常元符号c，A中元素$c^{\mathcal{A}}$的一个指派；以及对$\mathcal{L}$中每个n-元函数符号f，从A^n到A的一个n-元函数$f^{\mathcal{A}}$。

对于上面提到的几种只含一个二元谓词的语言，我们分别以$\mathcal{N}$，$\mathcal{Q}$或$\mathcal{Z}$为定义域来指定相应的结构。二元谓词的解释$P^{\mathcal{A}}$分别为$<$，$<$和$>$。如果增加二元函数符号f，那么必须在每种情形下，分别指定一个二元函数$f^{\mathcal{A}}$来解释它。在上述每个例子中，函数都可以选取：乘法、减法或最大值。

现在，我们着手在结构$\mathcal{A}$下解释$\mathcal{L}$中的公式，即对$\mathcal{L}$中的每个基本项，指出它命名了论域A中的哪个元素。

定义 4.2(基本项的解释)

(i)每个常元项c**命名**(name)元素$c^{\mathcal{A}}$。

(ii)如果$\mathcal{L}$中的项$t_1, \cdots, t_n$分别**命名**A中的元素$t_1^{\mathcal{A}}, \cdots, t_n^{\mathcal{A}}$，且$f$是$\mathcal{L}$的一个$n$-元函数符号，那么项$f(t_1, \cdots, t_n)$**命名**了$A$中的元素$f(t_1, \cdots, t_n)^{\mathcal{A}}=f^{\mathcal{A}}(t_1^{\mathcal{A}}, \cdots, t_n^{\mathcal{A}})$。(注意，$f^{\mathcal{A}}$是$A$上的一个$n$-元函数，$t_1^{\mathcal{A}}, \cdots, t_n^{\mathcal{A}}$是$A$中的元素，因而$f^{\mathcal{A}}(t_1^{\mathcal{A}}, \cdots, t_n^{\mathcal{A}})$实质上是$A$中的一个元素。)

继续上面的例子，考虑添加常元c和d到语言中，并将它们分别指派给结构中的元素$c^{\mathcal{A}}$和$d^{\mathcal{A}}$如下：

1)$c^{\mathcal{A}}=0$；$d^{\mathcal{A}}=1$。

2)$c^{\mathcal{A}}=1/2$；$d^{\mathcal{A}}=2/3$。

3)$c^{\mathcal{A}}=0$；$d^{\mathcal{A}}=-2$。

假设f在上面三个结构中都被解释为乘法，那么基本项$f(c, d)$与$f(d, f(d, d))$命名结构中的元素如下：

1)$(f(c, d))^{\mathcal{A}}=0$；$(f(d, f(d, d)))^{\mathcal{A}}=1$。

2)$(f(c, d))^{\mathcal{A}}=1/3$；$(f(d, f(d, d)))^{\mathcal{A}}=8/27$。

3)$(f(c, d))^{\mathcal{A}}=0$；$(f(d, f(d, d)))^{\mathcal{A}}=-8$。

对语言$\mathcal{L}$的结构$\mathcal{A}$，若有基本项来命名A中的每一个元素a，那么处理结构$\mathcal{A}$就很方便。如果在给定的语言$\mathcal{L}$的结构$\mathcal{A}$中，并非定义域中的每个元素都能由一个基本项所命名，那么我们扩充(expand)$\mathcal{L}$如下：对每个$a\in A$，添加新的常元c_a到$\mathcal{L}$中，得到语言$\mathcal{L}^A$，然后再把新添的常元解释为$c_a^{\mathcal{A}}=a$，从而得到$\mathcal{L}^A$的一个结构扩充了$\mathcal{A}$。因此在$\mathcal{L}^A$中，定义域A中的每一个元素都由一个常元命名。注意，通过这个办法，$\mathcal{L}$的每一个结构$\mathcal{A}$都变成了$\mathcal{L}^A$的一个结构，并且如果忽略常元c_a，那么$\mathcal{L}^A$的每一个结构都是$\mathcal{L}$的一个结构。

现在定义在给定语言$\mathcal{L}$的结构$\mathcal{A}$中，$\mathcal{L}$的语句φ何时为真。记之为$\mathcal{A}\vDash\varphi$。它的形式定义是通过对语句作归纳得到的。我们感兴趣的是量词情形。这里，假设有足够多的基本项来命名A中的每个元素。如果$\mathcal{L}$没有足够多的基本项，那么就用$\mathcal{L}^A$中的定义。因此，在下述定义

中，假设每一个 $a\in A$ 都由$\mathcal{L}$的一个基本项所命名。

定义 4.3 在结构$\mathcal{A}$下，每一个 $a\in A$ 都由$\mathcal{L}$的一个基本项所命名。归纳定义$\mathcal{L}$中的语句 φ 在结构$\mathcal{A}$下的**真值**(truth)如下(如果并非 A 的每一个元素都由$\mathcal{L}$中的一个基本项所命名，那么就利用$\mathcal{A}\vDash\varphi$ 在$\mathcal{L}^A$中的定义，来给出其在$\mathcal{L}$中的定义)：

(i)对原子语句 $R(t_1, \cdots, t_n)$，$\mathcal{A}\vDash R(t_1, \cdots, t_n)$当且仅当 $R^{\mathcal{A}}(t_1^{\mathcal{A}}, \cdots, t_n^{\mathcal{A}})$，即 A^n 上指派给 R 的关系 $R^{\mathcal{A}}$ 对由项 $t_1, \cdots, t_n$ 命名的那些元素保持成立。注意，由于 $R(t_1, \cdots, t_n)$ 是一个语句，所以 t_i 都是基本项，它们命名 A 中的特殊元素。

(ii)$\mathcal{A}\vDash\neg\varphi\Leftrightarrow\mathcal{A}\vDash\varphi$ 不成立。(也记作$\mathcal{A}\nvDash\varphi$。)

(iii)$\mathcal{A}\vDash(\varphi\vee\psi)\Leftrightarrow\mathcal{A}\vDash\varphi$ 或$\mathcal{A}\vDash\psi$。

(iv)$\mathcal{A}\vDash(\varphi\wedge\psi)\Leftrightarrow\mathcal{A}\vDash\varphi$ 且$\mathcal{A}\vDash\psi$。

(v)$\mathcal{A}\vDash(\varphi\rightarrow\psi)\Leftrightarrow\mathcal{A}\nvDash\varphi$ 或$\mathcal{A}\vDash\psi$。

(vi)$\mathcal{A}\vDash(\varphi\leftrightarrow\psi)\Leftrightarrow(\mathcal{A}\vDash\varphi$ 且$\mathcal{A}\vDash\psi)$或者$(\mathcal{A}\nvDash\varphi$ 且$\mathcal{A}\nvDash\psi)$。

(vii)$\mathcal{A}\vDash\exists v\varphi(v)\Leftrightarrow$对某个基本项 t，$\mathcal{A}\vDash\varphi(t)$。

(viii)$\mathcal{A}\vDash\forall v\varphi(v)\Leftrightarrow$对所有基本项 t，$\mathcal{A}\vDash\varphi(t)$。

注意，(ii)～(viii)中较长语句的真值(或满足性(satisfaction)，$\vDash$的通常叫法)总是由较短语句的真值定义。(vii)和(viii)需要如下假设：结构中的所有元素都由基本项所命名。

定义 4.4 给定某语言$\mathcal{L}$。

(i)$\mathcal{L}$的语句 φ 是**永真的**，即$\vDash\varphi$，如果它在$\mathcal{L}$的所有结构中都为真。

(ii)给定一个语句的集合 $\Sigma=\{\alpha_1, \cdots\}$，称 α 是 Σ 的**逻辑后承**，即 $\Sigma\vDash\alpha$，如果 α 在每个使得 Σ 的所有成员为真的结构中为真。

(iii)一个语句集合 $\Sigma=\{\alpha_1, \cdots\}$ **可满足**，如果存在一个结构$\mathcal{A}$，使得 Σ 的所有成员在其中为真。这样的结构叫做 Σ 的**模型**。若 Σ 没有模型，则其**不可满足**。

注意，这里定义的真值只针对语句，即不含自由变元的公式。若 $\varphi(v)$中有自由变元 v，则该公式在一个结构$\mathcal{A}$下没有固定的含义。它只是表示 A 上的一个 n-元谓词，其中 $n>0$。因此，我们不说它为真或为假。对于带有自由变元的公式，有一个概念与语句的真值相类似，即永真性。

定义 4.5 语言$\mathcal{L}$中带有自由变元 $v_1, \cdots, v_n$ 的公式 φ **在$\mathcal{L}$的一个结构$\mathcal{A}$下是永真的**(valid in a structure $\mathcal{A}$ for $\mathcal{L}$，记作$\mathcal{A}\vDash\varphi$)，如果 φ 的**全称闭包**(universal closure)在$\mathcal{A}$下为真。这里 φ 的全称闭包是指，对 φ 中每一个自由变元 v_i，把 $\forall v_i$ 放到 φ 的前面而得到的语句 $\forall v_1\forall v_2\cdots\forall v_n\varphi$。$\mathcal{L}$的公式 φ 是**永真的**，如果它在$\mathcal{L}$的每一个结构下都是永真的。

只要结构$\mathcal{A}$中的每一个元素都由一个基本项所命名，$\mathcal{A}$中的永真性定义就等价于 φ 的每一个基本实例在$\mathcal{A}$中为真，即对$\mathcal{L}$中的所有基本项 $t_1, \cdots, t_n$，$\mathcal{A}\vDash\varphi(t_1, \cdots, t_n)$。另外也要注意，由于语句不含任何自由变元，因此一个语句在某个结构中为真，当且仅当它在该结构中是永真的。

提醒 对语句 φ 和结构$\mathcal{A}$，φ 和$\neg\varphi$ 之中必有一个在$\mathcal{A}$下为真(而另一个为假)。然而这并不是说对任意的公式 ψ，ψ 和$\neg\psi$ 之中必有一个在$\mathcal{A}$下永真。可能出现 ψ 的某些基本实例为真而其余基本实例为假的情形。类似地，可以找到一个语句，使得它和它的否定均非永真，即在某些结构下该语句及其否定中一定有一个是永真的，而在其余结构下则不然。

定义 4.6 一个由带有自由变元的公式组成的集合 Σ **可满足**，如果存在一个结构，使得

*σ 中的所有公式在该结构下永真(即它们的全称闭包为真)。这样的结构叫做 Σ 的***模型***。若 Σ 没有模型，则其***不可满足***。*

例 4.7 考虑一个语言$\mathcal{L}$，它由二元关系符号 R 以及常元 c_0，c_1，c_2，…指定。下面给出$\mathcal{L}$的两种可能的结构，它们对应了语言$\mathcal{L}$的两种不同的解释。

(i)令定义域 A 为自然数集，$R^{\mathcal{A}}$ 为通常关系 <，$c_0^{\mathcal{A}}=0$，$c_1^{\mathcal{A}}=1$，…。语句$(\forall x)(\exists y)R(x, y)$是说，对每一个自然数，总存在一个比它大的自然数。因此，在此结构下该语句为真。

(ii)令$\mathcal{A}$的定义域为有理数集 $Q=\{q_0, q_1, \cdots\}$，$R^{\mathcal{A}}$ 仍为通常关系 <，$c_0^{\mathcal{A}}=q_0$，$c_1^{\mathcal{A}}=q_1$，…。语句$(\forall x)(\forall y)(R(x, y)\to(\exists z)(R(x, z)\wedge R(z, y)))$在此结构下为真(即有理数稠密)。但它不是永真的，因为它在(i)中的结构下(自然数的情形)为假。

提醒 无论是在谓词逻辑的语法还是语义里，我们都没有设置任何特殊的或者保留谓词符号来表示相等关系。换句话说，从我们的定义出发并不能把某个特殊谓词，比如“=”，解释为真正的相等。我们在定义 2.1(关于语言的定义)以及定义 4.3(关于真值的定义)中，均回避了这一扩充，因为它不能与消解定理证明、逻辑式程序设计以及 PROLOG 很好地兼容。对于这一扩充可能带来的其他影响，参见第七节习题 2~3。第三章第五节考察了这种带有专门相等谓词的谓词逻辑(同时给出了一种方法，在不含这种专门谓词的条件下推出相等性质)。该部分给出的语法和语义现在可以参阅，而对于带有专门相等谓词的谓词逻辑，其可靠性及完全性证明只是本章第七节相应证明的简单修改。

我们已经定义了谓词逻辑的语义，现在可以确切地断言：我们不需要命题的概念。事实上，命题逻辑能够可靠地嵌入到谓词逻辑。

定理 4.8 *令 φ 是谓词逻辑中的一个开公式(即不含量词)。如果把 φ 的每一个原子子公式当作一个命题字母，那么可以把 φ 看作命题逻辑中的公式 φ'。由此对应关系知，φ 是谓词逻辑中的一个永真公式，当且仅当 φ'在命题逻辑中永真。*

证明 留作习题 8~11。 □

现在我们既有了谓词逻辑的语法，又有了它的语义。按照命题逻辑的研究思路，下一步要做什么应该很清楚了。我们必须给出谓词逻辑中的证明方法，然后像命题演算那样去证明可靠性和完全性定理。但是在此之前，先来考虑语义在 PROLOG 中的应用(实质上是一种特殊化)。

备注 我们经常利用向量符号$\vec{x}$，$\vec{t}$和$\vec{c}$分别表示变元序列、项序列和常元序列。

习题

1. 令$\mathcal{L}$包含常元 c、二元函数 f 以及一元谓词 P。给出$\mathcal{L}$的两个结构，使得$\forall x\, P(f(x, c))$在它们中分别为真和假。
2. 证明：$\forall x(p(x)\to q(f(x)))\wedge\forall x\, p(x)\wedge\exists x\,\neg q(x)$可满足。
3. 给出一个不可满足的语句的例子。
4. 对包含常元符号 0 和 1、二元谓词 < 以及二元函数符号 + 的语言，定义一个结构，使得 $x+1<x$ 在其中永真而 $x+x<x$ 非永真。说明该结构为何具有我们所要求的性质。
5. 证明$\mathcal{A}\vDash\neg\exists x\varphi(x)\Leftrightarrow\mathcal{A}\vDash\forall x\,\neg\varphi(x)$。如果 φ 有除 x 之外的自由变元，那么该结论是否仍然成立？
6. 证明：对任何语句 ψ，$\mathcal{A}\vDash(\psi\to\exists x\varphi(x))\Leftrightarrow\mathcal{A}\vDash\exists x(\psi\to\varphi(x))$。如果 x 在公式 ψ 中自由，那么该结论是否仍然成立？
7. 证明：对任何语句 ψ，$\mathcal{A}\vDash(\exists x\varphi(x)\to\psi)\Leftrightarrow\mathcal{A}\vDash\forall x(\varphi(x)\to\psi)$。如果 x 在公式 ψ 中自由，那么该结论是

否仍然成立？

8. **常元定理**：令 $\varphi(\vec{x})$ 是语言 $\mathcal{L}$ 中的一个公式，$\vec{x}$ 是自由变元的一个序列。又令 $\vec{c}$ 是(不在 $\mathcal{L}$ 中的)新常元的一个序列。证明 $\varphi(\vec{x})$ 是永真的，当且仅当 $\varphi(\vec{c})$ 永真。
9. 证明定理 4.8 的公式不含自由变元的情形。(提示：把 φ 或 $\neg\varphi$ 的模型转化成使 φ' 或 $\neg\varphi'$ 为真的指派。)
10. 结合习题 8 和 9 来证明定理 4.8。

第五节 PROLOG 程序解释

本节旨在将上一节中的思想及定义特殊化，从而解释子句形式以及含有自由变元的 Horn 公式的语义。下面从研究完全 PROLOG 程序的语义开始。

子句形式及 PROLOG 模式的语法和命题情形(第一章定义 10.4)下的语法基本相同，唯一的差别在于，这里的文字可以是任何原子公式或其否定。注意，在 PROLOG 的执行中，统一使用(初始)大写字母表示变元(的名字)，小写字母表示谓词、常元及函数(的名字)。

定义 5.1 (子句表示，clausal notation)

(i) **文字**是指原子公式或其否定。原子公式叫做**正文字**，其否定叫做**负文字**。

(ii) **子句**是指文字的一个有穷集合。

(iii) Horn **子句**是指至多包含一个正文字的子句。

(iv) **程序子句**是指恰好含有一个正文字的子句。含有负文字的程序子句叫做**规则**，不含负文字的程序子句叫做**事实**。

(v) **目标子句**是指不含正文字的子句。

(vi) **公式**是指子句的一个(不必有穷的)集合。

规则与事实的 PROLOG 表示和其在命题情形下完全一样。

定义 5.2 (PROLOG 表示)

(i) 在 PROLOG 中，事实 $\{p(\vec{X})\}$ 是由出现在 PROLOG 程序中的如下单个正文字 $p(\vec{X})$ 组成：

$$p(\vec{X}).$$

(ii) 出现在 PROLOG 程序中的规则 $C=\{p(\vec{X}),\ \neg q_1(\vec{X},\ \vec{Y}),\ \cdots,\ \neg q_n(\vec{X},\ \vec{Y})\}$ 如下：

$$p(\vec{X}) \text{ :- } q_1(\vec{X},\vec{Y}),\cdots,q_n(\vec{X},\vec{Y}).$$

(iii) 对(ii)中的规则 C，我们把 $p(\vec{X})$ 叫做 C 的**目标**或者**头**，把 $q_1(\vec{X},\ \vec{Y})$，…，$q_n(\vec{X},\ \vec{Y})$ 叫做 C 的**子目标**或者**体**。在头-体术语系统下，连接头和体的符号":-"叫做**颈**。

(iv) (PROLOG) **程序**是指仅包含程序子句(即规则与事实)的公式(子句的集合)。

子句及公式在 PROLOG 中的含义与在命题情形下基本相同，唯一的差别在于，这里我们必须解释如何处理自由变元。每个子句都被解释为其元素析取的全称闭包。因此，$C_1=\{q(X,\ Y),\ r(Y)\}$ 即指 $\forall X\forall Y[q(X,\ Y)\vee r(Y)]$。在此释义下，由 $p(X)\text{:-}q_1(X,\ Y),\ \cdots,\ q_n(X,\ Y)$ 给出的规则 C(写成子句形式即为 $C=\{p(X),\ \neg q_1(X,\ Y),\ \cdots,\ \neg q_n(X,\ Y)\}$)是指 $\forall X\forall Y[p(X)\vee\neg q_1(X,\ Y)\vee\cdots\vee\neg q_n(X,\ Y)]$。重复应用第四节习题 8 及第一章第三节习题 2，可以得知它等价于 $\forall X[\exists Y(q_1(X,\ Y)\wedge\cdots\wedge q_n(X,\ Y))\rightarrow p(X)]$。(稍后我们分析几个例子，来看一看这种等价是如何建立起来的。)因此，C 真正具体化了一条规则：对任何 X，如果存在一个 Y，使得 $q_1(X,\ Y)$，$q_2(X,\ Y)$，…，$q_n(X,\ Y)$ 均为真(已经验证)，那么

$p(X)$也为真(已经验证)。

公式S被解释为其子句的合取。因此，如果$S=\{C_1, C_2\}$，其中C_1如上所示，$C_2=\{q(X, Y), m(Y)\}$，那么S的含义与$\forall X\forall Y[q(X, Y)\vee r(Y)]\wedge\forall X\forall Y[q(X, Y)\vee m(Y)]$一样。特别地，如果公式$S$是一个PROLOG程序，那么它等价于如下一个列表：该列表由形如$\forall Xp(X)$的全称事实及诸如上述C的规则共同组成。PROLOG的执行就是从这样的由事实与规则组成的列表出发，进行演绎。

注意，在描述公式的含义时，合取之前的每个子句都是全称封闭的。这一约定的重要性将会在考虑谓词演算的消解时体现出来。为了避免混淆，我们在各个子句中使用不同的变元。(这对应了所谓的标准化变元分离(standardizing the variables apart)(见第十三节))。后面(第九节)将证明，对我们的语言添加新的函数符号之后，谓词演算中的每个语句都等价于定义5.1意义下的公式。(这一结果类似于谓词演算中的CNF。)现在给出一个关于这种转化的例子，对于由子句形式表示的公式，直接处理其语法和语义。此外，对这种子句形式的公式，其结构表示以及解释都是由上述转化指定。

***骑士移动：一个例子**

通过考虑棋盘上骑士移动的信息表示，简要地检验PROLOG中使用变元所带来的表述能力的增强。我们用从1到8的数对来标识棋盘的方格，如图28所示。

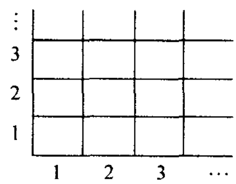


图 28

我们希望我们的语言包含常元符号1，2，…，8。我们语言中的基本谓词是4-元的，ktmove(X_1, X_2, X_3, X_4)。(此处，希望通过第三节引进的列表概念来代替PROLOG中二元数对的实际表示，即利用$\{1, \cdots, 8\}$上的4-元谓词来表示移动。)"ktmove(X_1, X_2, X_3, X_4)"是指骑士可以从位置$\langle X_1, X_2\rangle$移至$\langle X_3, X_4\rangle$。表示相关数据的一个办法就是列举所有的事实：

ktmove(1,1,2,3).
ktmove(1,1,3,2).
…

但是这个列表太长了(336个事实)。并且如果再多询问一点，这个情形很快就会变得难以忍受。假设还需要另外一个谓词2ktmove(X_1, X_2, X_3, X_4)，它是指骑士从$\langle X_1, X_2\rangle$两步移至$\langle X_3, X_4\rangle$。我们仍然要输入一个很长的事实列表：

2ktmove(1,1,1,1).
2ktmove(1,1,3,5).
…

如果限制在命题逻辑，则只能做到这些。这是只含数据库的情形。一旦有了变元，就能引入规则，从而大大压缩数据表示。2ktmove即是一个明显的例子：2ktmove(X_1, X_2, X_3, X_4)成立，如果存在Y_1, Y_2，使得ktmove(X_1, X_2, Y_1, Y_2)与ktmove(Y_1, Y_2, X_3, X_4)成立。在谓词逻辑中可以将此结果或定义表述如下：

$$\forall X_1\forall X_2\forall X_3\forall X_4[\exists Y_1\exists Y_2(\text{ktmove}(X_1,X_2,Y_1,Y_2)\wedge\text{ktmove}(Y_1,Y_2,X_3,X_4))\rightarrow\text{2ktmove}(X_1,X_2,X_3,X_4)]。(\#)$$

在第九节中将引入一般的方法，把谓词演算中所有这样的语句转化成等价的子句形式(或PROLOG程序)。现在暂且用一种特殊方法来分析。首先，和命题逻辑中构造CNF一

样，用$\neg$和$\vee$来消除蕴涵联结词，得到(#)的等价语句：

$$\forall X_1 \forall X_2 \forall X_3 \forall X_4[\neg(\exists Y_1 \exists Y_2)(\mathrm{ktmove}(X_1,X_2,Y_1,Y_2) \wedge \mathrm{ktmove}(Y_1,Y_2,X_3,X_4)) \vee 2\mathrm{ktmove}(X_1,X_2,X_3,X_4)]。$$

然后应用第四节习题5 $\neg\exists z\varphi$ 的等价形式 $\forall z\neg\varphi$ 以及第一章第三节习题2中的德·摩根定律，得到：

$$\forall X_1 \forall X_2 \forall X_3 \forall X_4[\forall Y_1 \forall Y_2[\neg\mathrm{ktmove}(X_1,X_2,Y_1,Y_2) \vee \neg\mathrm{ktmove}(Y_1,Y_2,X_3,X_4)] \vee 2\mathrm{ktmove}(X_1,X_2,X_3,X_4)]。$$

最后得到与(#)等价的子句形式：

$$\forall X_1 \forall X_2 \forall X_3 \forall X_4 \forall Y_1 \forall Y_2[\neg\mathrm{ktmove}(X_1,X_2,Y_1,Y_2) \vee \neg\mathrm{ktmove}(Y_1,Y_2,X_3,X_4) \vee 2\mathrm{ktmove}(X_1,X_2,X_3,X_4)]。$$

(以上两个语句的语义等价关系是比较清楚的。)

因此，在谓词演算下的规则(#)的子句形式即为：

$$\{\neg\mathrm{ktmove}(X_1,X_2,Y_1,Y_2),\neg\mathrm{ktmove}(Y_1,Y_2,X_3,X_4), 2\mathrm{ktmove}(X_1,X_2,X_3,X_4)\}。$$

这是一个 Horn 子句，写成 PROLOG 形式即为：

$$2\mathrm{ktmove}(X_1,X_2,X_3,X_4) :\text{-}\ \mathrm{ktmove}(X_1,X_2,Y_1,Y_2), \mathrm{ktmove}(Y_1,Y_2,X_3,X_4).$$

至此，对 PROLOG 中形如"$p(\vec{X}):-q_1(\vec{X},\ \vec{Y}),\ \cdots,\ q_n(\vec{X},\ \vec{Y})$"的程序子句，我们得到了一个关于它的一般解释的例子。这条规则是说，对子句目标(头)$p(\vec{X})$中变元$\vec{X}$的每一个选择，p在$\vec{X}$中成立(**成功**)，如果存在$\vec{Y}$，使得所有子目标子句(体)$q_1(\vec{X},\ \vec{Y})$，$\cdots$，$q_n(\vec{X},\ \vec{Y})$成立(**成功**)。在我们的情形中该子句是说，我们可以从$\langle X_1,\ X_2\rangle$，两步移至$\langle X_3,\ X_4\rangle$，如果存在一个$\langle Y_1,\ Y_2\rangle$，使得我们可以从$\langle X_1,\ X_2\rangle$一步移至$\langle Y_1,\ Y_2\rangle$，再从$\langle Y_1,\ Y_2\rangle$一步移至$\langle X_3,\ X_4\rangle$。

下面再来看看怎样通过引入规则，把 ktmove 程序表示的大小从 336 个子句减少至一个更易于处理的数目。

一种方法是引入对称性规则，它使得我们能从少许基本移动导出骑士的所有移动。对称性本身就是这样一个明显的例子。

(S1) $\mathrm{ktmove}(X_1,\ X_2,\ X_3,\ X_4):-\mathrm{ktmove}(X_3,\ X_4,\ X_1,\ X_2).$

记住，这条规则是说(对任何X_1，X_2，X_3，X_4)，如果一个骑士可以从$\langle X_3,\ X_4\rangle$移至$\langle X_1,\ X_2\rangle$，那么它就可以从$\langle X_1,\ X_2\rangle$移至$\langle X_3,\ X_4\rangle$。引入这条规则即可把我们的数据库砍掉一半。其他可能的规则如下：

(S2) $\mathrm{ktmove}(X_1,\ X_2,\ X_3,\ X_4):-\mathrm{ktmove}(X_1,\ X_4,\ X_3,\ X_2).$

(S3) $\mathrm{ktmove}(X_1,\ X_2,\ X_3,\ X_4):-\mathrm{ktmove}(X_2,\ X_1,\ X_4,\ X_3).$

(S4) $\mathrm{ktmove}(X_1,\ X_2,\ X_3,\ X_4):-\mathrm{ktmove}(X_3,\ X_2,\ X_1,\ X_4).$

(验证这些规则事实上是国际象棋中骑士走法的正确规则)。如此一来，我们只需少许基本移动，再加上上述程序子句，即可准确定义谓词 ktmove。(这在如下意义下是正确的：任何满足所有这些事实与规则的结构都会准确给出骑士的合法移动，这里合法移动是指使得谓词 ktmove 成立的取自常元$\{1,\ \cdots,\ 8\}$的 4-元组。)程序执行的正确性要用消解型定理来证

明，我们把它放在后面处理。

另一个方法就是尝试用坐标上的算术操作来定义“ktmove”，即在某种意义下抓住规则，像通常所做的那样：骑士可以从$\langle X_1, X_2\rangle$移至$\langle X_3, X_4\rangle$，如果该移动在两个方向上的坐标值变化分别为1和2，即

(A0)ktmove(X_1, X_2, X_3, X_4)，如果$|X_1-X_3|+|X_2-X_4|=3$。

(我们还必须确保这两个坐标不同。这依赖于定义适当的算术操作。特别地，$|X_1-X_3|$不得为0。)现在PROLOG中已有很多内部算术操作与谓词，但是要想准确地理解它们是如何被使用的，还需要更多地了解程序的执行过程。到目前为止，我们希望避免使用内部谓词。不过我们可以把所需要的算术添加到程序定义当中。(注意不要使用内部谓词的保留名字。)

接下来，用一个数据库在数据{1, …, 8}上定义谓词“后继”：

```
suc(1,2).
suc(2,3).
   ⋮
suc(7,8).
```

然后，可以用如下规则定义加法的缩减版本：

(A1)add(X, 1, Z):- suc(X, Z).

(A2)add(X, Y, Z):- suc(Y_1, Y), suc(Z_1, Z), add(X, Y_1, Z_1)

在此基础上直接定义$|X_1-X_2|=Y$如下：

(A3)absolute_difference(X_1, X_2, Y):- add(X_1, Y, X_2).

(A4)absolute_difference(X_1, X_2, Y):- add(X_2, Y, X_1).

(这些规则只是刚刚满足了我们的需要，因为我们只对缩减后的操作感兴趣，即作用在{1, …, 8}上的那些操作。它们并未准确定义所有整数上的操作。稍后，我们会更多地谈到程序的结构解释。)

到目前为止，我们一直在考虑PROLOG程序中子句的含义，这是通过不断向一个包含该程序的文件发出“询问”来完成的。现在我们必须解释在“?”提示符处所输入目标子句的语义。例如，“?- $p(X_1, X_2)$, $q(X_2, X_3)$.”意指“是否存在个体a_1, a_2, a_3，使得$p(a_1, a_2)$与$q(a_2, a_3)$”成立。PROLOG不仅负责对该问题回答是或否，而且如果回答为“yes”，它会给出实例来验证该问题，即给出实际的项a_1, a_2和a_3(从而命名了个体)，使得$p(a_1, a_2)\wedge q(a_2, a_3)$成立。(正如第一章定义10.4命题逻辑时的情形，在得到一个回答之后输入“;”，再寻找另一个答案。反复执行这一过程，直至没有答案产生，此时PROLOG回答“no”。在搜寻答案的过程中输入“回车”同样会导致终止。)

和命题逻辑的情形一样，PROLOG通过如下方式来执行搜索证据a_1, a_2, a_3的过程：添加目标子句$G=\{\neg p(X_1, X_2), \neg q(X_2, X_3)\}$到当前程序$P$，然后判定它是否为不可满足公式。下面列出该公式不同语义的等价形式，来帮助观察搜索程序是如何对我们的问题给出答案的。首先，子句G意指$\forall X_1\forall X_2\forall X_3[\neg p(X_1, X_2)\vee\neg q(X_2, X_3)]$。如果将其加到程序$P$中得到了一个不可满足的公式$P\cup\{G\}$，那么它的否定是$P$的一个逻辑后承(和在第一章引理10.6中一样，通过定义来验证)。因此

$$P\vDash\neg\forall X_1\forall X_2\forall X_3[\neg p(X_1,X_2)\vee\neg q(X_2,X_3)]$$

正如上面所见(以及第四节习题5)，它等价于

$$P \vDash \exists X_1 \exists X_2 \exists X_3 [p(X_1, X_2) \wedge q(X_2, X_3)]$$

PROLOG 的执行，试图通过生成 $P \cup \{G\}$ 的消解反驳，来建立这种后承关系。(在本章第十三节及第三章将定义谓词演算的消解反驳。)该消解反驳的证明过程生成了一个副产品：例如在本例中，它生成了证据 a_1，a_2，a_3，通过提供一个从 P 出发的证明 $p(a_1, a_2) \wedge q(a_2, a_3)$，来表明 $P \cup \{G\}$ 不可满足。从消解定理证明的角度来看，这些证据仅是证明过程的副产品；而从编程的角度来看，它们就是本质结果。这些证据是程序的输出，即问题的答案。

PROLOG 是通过逻辑表示数据，这种方式使得其输入与输出之间存在一种罕见的对称。在询问问题时，我们可以把变元放到谓词中的任何地方。因此，简单谓词 add(X, Y, Z)不仅在输入"? - add(a, b, Z)."时给出 $a+b$，而且在输入"? - add(a, Z, b)."时(至少在 $b>a$时)给出 $b-a$。因此，对于那些很难从一个简单数据库出发得到答案的技巧性问题，我们可以尝试用单个的 PROLOG 程序来解决。如果询问骑士是否能够从 $\langle a, b\rangle$ 三步移至 $\langle c, d\rangle$，对比以下两种解决方案：一是利用上述 PROLOG 程序之一，一是用其他语言明确写出这个程序，即给出一个列有骑士所有移动的数据库。搜索安排及搜索顺序都是自动完成。我们稍后会再来研究上述的可逆性和搜索过程。

习题

1. 验证对称规则(S1)~(S4)是合理的。(提示：可以应用 ktmove(A0)的算术定义来验证。)
2. (用自然语言)解释规则(A1)~(A2)的含义，并解释它们是如何准确表示结构{1，…，8}上的加法的。
3. (用自然语言)解释规则(A3)~(A4)的含义，并解释它们是如何准确表示结构{1，…，8}上的绝对差的。
4. 假设 suc(X, Y)以某种方式正确地定义在所有自然数上，即 suc(n, m)为真，当且仅当 $n+1=m$。

   a)子句(A1)~(A2)是否还能准确定义加法？

   b)子句(A3)~(A4)是否还能准确定义绝对差？

   假设 suc(X, Y)定义整数上的"后继"。那么定义在整数上的子句(A1)~(A2)和子句(A3)~(A4)的关系如何？
5. 假设某个语言包含常元 c，一元函数符号 $s(X)$以及三元谓词符号 $a(X, Y, Z)$。写出一个 PROLOG 子句的集合，使得"a"在下述意义下定义了加法：$a(s^n(c), s^m(c), s^t(c))$是该程序的后承，当且仅当 $n+m=t$。($s^n(c)$是 $s(\cdots(s(c))\cdots)$的简写，这里 s 总共出现 n 次。)
6. 证明：所有 PROLOG 程序都是可满足的。

   下面所设计的(以及以后的一些)问题要用到我们网上提供的数据库。该数据库是由《旧约-历代记》(Chronicles,《希伯来圣经》(Hebrew Bible)的最后一本书)的前几章给出的家谱图组成。那里的信息仅包含男性后代。(实际上母亲和孩子之间也有少量信息，但是太零散了，使得蕴涵关系变得无用。)数据库中的信息是通过谓词"fatherof(a, b)"来记录的。因此，全部由(许多)事实组成的文件可以输入如下：

```
fatherof(adam, seth).
fatherof(abraham, isaac).
fatherof(isaac, jacob).
fatherof(isaac, esau).
```

   在习题 7 和 8 中，假设上述类型的文件是唯一可用的(例如，在定义祖父时，不必考虑母亲一方的祖先，因为这类信息不可用。)

   我们在附录 B 中打印出了上述数据库。如果读者无法在线连到该数据库或者相似的数据库，那么在回答下列问题时只需要写出一个 PROLOG 程序，并保证根据对本节所述事实与规则的解释，该程序是语义正确的。类似地，如何从程序中获取要求的信息亦能得以描述。
7. 祖先：

a)写一个程序来定义“grandfatherof”。

b)找出 nimrod、lud 和 joktan 的祖父。

c)利用上面的程序找出 noah 的一个孙子；并找出其所有孙子(利用交互式问答，每收到一个回答就输入一个分号，直至没有回答出现)。

d)写一个程序来定义“greatgrandfatherof”。

e)找出 shem 和 canaan 的曾祖父。

f)利用上面的程序找出 abraham 的一个曾孙；并找出其十个曾孙。

g)写一个程序来定义“ancestorof”。

h)找出 shem 的三个祖先。

8. 叔父：

a)写一个程序来定义“uncleof”。

b)找出 nimrod、lud 和 joktan 的叔父。

c)利用上面的程序找出 shem 的一个侄子；并找出其所有侄子(利用交互式问答，每收到一个回答就输入一个分号，直至没有回答出现)。

d)写一个程序来定义“granduncleof”(注意曾叔父是曾祖父的兄弟)。

e)找出 shelah 和 canaan 的曾叔父。

f)利用上面的程序找出 ham 的曾侄子；并找出其八个曾侄子。

注意在本题中可以利用带有不等谓词($X\neq Y$)的 PROLOG 版本。

第六节 证明：完全系统表

现在描述一个用于构造谓词逻辑中语句证明的系统。和命题逻辑一样，这些证明是标签二叉树，叫做表。树上的标签是*标号语句*(signed sentence)(即以 T 或 F 开始的语句，意思是说，按照分析需要分别假定相关语句为真或为假)。我们仍然把这些标签叫做表的*表值*。形式地，归纳定义谓词逻辑中的表如下：首先指定一些特定的(标签二叉)树作为表(所谓的原子表)，然后给出一个扩展规则来定义更复杂的表。证明过程的要领即是，从某个诸如 $F\alpha$ 的标号语句开始，以其为根将其分解到各分支，如此继续下去，最终说明任何一个分解都导致矛盾。于是得出结论：推翻了原先 α 为假的假设，从而得到 α 的一个证明。

联结词的分解和在命题逻辑中一样。分解的要领是：如果某个语句被正确的符号标识(T 或 F)，那么它的直接后继当中至少有一个也被正确的符号标识。一个新的问题是如何处理量词。例如，考虑 $T\exists x\varphi(x)$，直观的分析是存在一个 x，使得 $\varphi(x)$ 成立。要求 $\varphi(x)$ 成立是为了给出这样一个 x。仅仅提供这么一个 x。提供这么一个证据意味着指定一个基本项 t，并断言 $\varphi(t)$ 为真。因此，首要的问题就是，要有足够多的基本项以满足我们的需要。如果是从任意语言$\mathcal{L}$开始，通过添加未出现于$\mathcal{L}$中的常元符号 c_0，c_1，c_2，…，把$\mathcal{L}$扩充成$\mathcal{L}_C$。令 A 是$\mathcal{L}_C$中的任一原子语句，α，β 是$\mathcal{L}_C$中的任意语句。对于用来分析语言$\mathcal{L}$中语句的那些表，其归纳定义的基本情形是由图 29 所示的(标签二叉)树开始的。

直观上，容易理解为何要求情形 7b 和 8a 所引入的常元必须是“新”的。这里表的起始点是如下断言：存在带有某种性质的 x。只要对 c 没有更优先的要求，则可以断言 c 就是这样的一个 x。另一方面，如果已经有其他断言涉及 c，那么我们就没有权利假设带有这些性质的元素仍能成为新断言的证据。“新”的精确语法含义是与表(以标号语句为标签的二叉树)的归纳定义同时定义出来的。

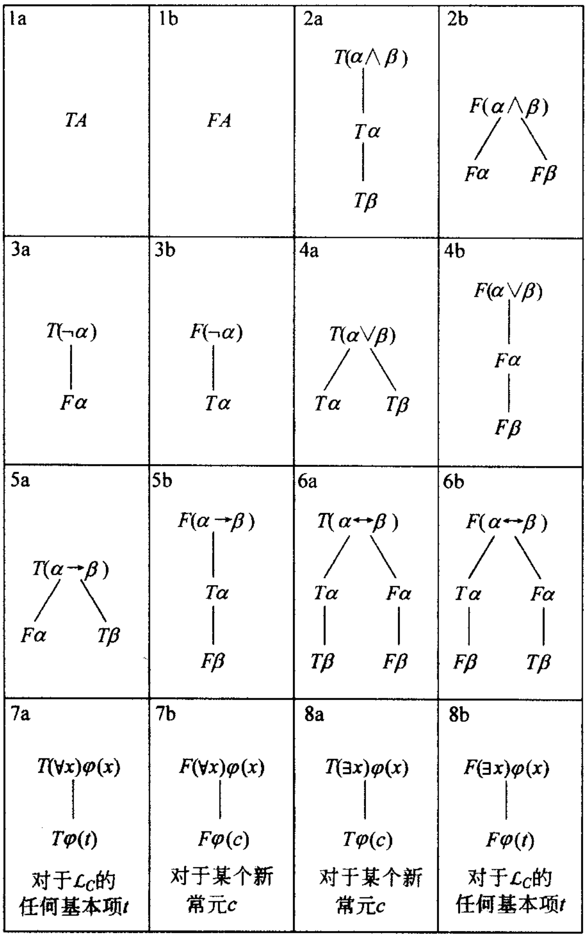


图 29

定义 6.1 我们把表归纳地定义成以($\mathcal{L}_C$中的)标号语句(叫做表值)为标签的二叉树:

(i)所有的原子表都是表。情形 7b 和 8a 要求 c 是新的，在这里仅指 c 是从$\mathcal{L}$到$\mathcal{L}_C$所添加的那些常元 c_i 之一(因此它不出现在 φ 中)。

(ii)如果 τ 是一个有穷表，P 是 τ 上一条路径，E 是 τ 中出现在 P 上的一个表值，并且 τ'是从 τ 出发通过将根为表值 E 的原子表连接到 τ 中路径 P 的末端得到的，那么 τ'也是一个表。情形 7b 和 8a 要求 c 是新的，在这里指 c 是不出现在 P 上任何表值中的 c_i 之一。(在实际操作中，选取一个未出现于 τ 中任何节点的新常元，要比书本表述简单得多。)

(iii)如果 τ_0 是一个有穷表，且 τ_0，τ_1，…，τ_n，…是一个表的序列，使得对每个 $n\geqslant 0$，τ_{n+1}是由 τ_n 应用(ii)构造而来，那么 $\tau=\bigcup\tau_n$ 也是一个表。

提醒 谓词逻辑中关键的一点在于，对于第(ii)条里的表值 E，当其对应的原子表被添加到 P 上时，表值 E 重复出现一次(至少在 7a 和 8b 的情形下)。当我们分析这些情形下所需的行为以及完成表的定义(定义 6.7)时，E 重复出现的原因就比较清楚了。

接下来，定义谓词逻辑中语句的表证明。首先需要注意的是，在大多数情形下不会只证明一个单独的语句，通常证明带有各种假设或公理的语句。此过程的语义表述嵌套在定

义 4.4逻辑后承的概念之中。为了描述相应的证明论概念，需要定义谓词逻辑中从前件出发的表及证明，这与第一章第六节命题逻辑的情形类似。所需的修改也类似于第一章第六节命题逻辑中的定义所进行的修改。关键的变化集中在从语句集合 S 出发的表的定义当中。基本思路是假设 S 中的所有语句为真。因此，除了普通表的形成规则，我们还可以随时断言 S 中的任一语句为真。在形式化定义中，我们通过添加一条新的规则来表述这一思想。

在本节的剩余部分，令 S 为语言$\mathcal{L}$的一个语句集合。通常将 S 中的元素称作前件。

定义 6.1(续) **从 S 出发的表**(tableau from S)。从 S 出发的表的定义比普通表的定义多了下面一条形成规则：

(ii′)如果 τ 是一个从 S 出发的有穷表，φ 是一个从 S 出发的语句，P 是 τ 上的一条路径，并且 τ'是从 τ 出发通过连接 $T\varphi$ 到路径 P 的末端得到的，那么 τ'也是一个从 S 出发的表。

现在同时定义有关从 S 出发的表和普通表的概念。如下面备注所示，“(从 S 出发的)”代表从 S 出发的表的情形。

备注 由定义容易看出，每一个(从 S 出发的)表 τ 都是(从 S 出发的)表的有穷或无穷序列 τ_0，τ_1，…，τ_n，…的并集，这里 τ_0 是一个原子表并且每个 τ_{n+1}都是 τ_n 应用(ii)(或(ii′))得到的。从现在开始，始终假设所有(从 S 出发的)表都被表示成这种并集的形式。

定义 6.2 **(从 S 出发的)表证明**：令 τ 是一个表，P 是 τ 中一条路径。

(i)P 是**矛盾的**，如果对某个语句 α，$T\alpha$ 和 $F\alpha$ 同时作为 P 上节点的标签出现。

(ii)τ 是**矛盾的**，如果 τ 中所有路径都是矛盾的。

(iii)τ 是 α 的一个**(从 S 出发的)证明**，如果 τ 是一个(从 S 出发的)有穷矛盾表且根节点标签为 $F\alpha$。如果存在 α(从 S 出发)的证明 τ，那么我们说 α 是(从 S 出发)可证的，并记作$\vdash\alpha$($S\vdash\alpha$)。

(iv)S 是**不相容的**，如果对某个语句 α，存在 $\alpha\wedge\neg\alpha$ 的一个从 S 出发的证明。

注意，如果存在一个以 $F\alpha$ 为根的(从 S 出发的)矛盾表，那么就有一个以 $F\alpha$ 为根的(从 S 出发的)有穷矛盾表，即存在 α 的一个(从 S 出发的)证明。只要某条路径上出现矛盾，就终止该路径。现在，由于每条路径都是有穷的，根据库尼西引理，整棵树就是有穷的。因此，对任何语句证明的存在性，“证明必须是有穷表”这一附加要求并未起到任何作用。从另一个角度来看，对于表的定义(定义 6.1)中第(ii)条提到的路径 P，我们可以要求其为非矛盾路径，而且不会影响证明的存在性。

在描述完成表的形式定义以及完全系统表的构造之前，先来看一些谓词逻辑中表证明的例子，这对后面的工作有很大的启发意义。注意，仍然采用表的缩写形式：除非要处理原子表 7a 或 8b 的情形，我们不重复列出将要分解的表值。

例 6.3 假定我们要验证公式$((\forall x)\varphi(x)\rightarrow(\exists x)\varphi(x))$的永真性。形成的表如图 30 所示。

对于最后一个表值，为了导出矛盾，选择使用前面已经出现过的常元 c。之所以能够这样做，是因为$\forall x\varphi(x)$的原子表允许使用任意常元。

下面的例子同样生成一个矛盾表。

例 6.4 如图 31 所示。

在实际应用中，我们通常首先扩展需要引入新的项的原子表，然后再转向那些任何基本项都能用得上的原子表，这样做更为高效。

例 6.5 如图 32 所示。

- $F[(\forall x)\varphi(x)\rightarrow(\exists x)\varphi(x)]$
- $T(\forall x)\varphi(x)$
- $F(\exists x)\varphi(x)$
- $F\varphi(c)$
- $T(\forall x)\varphi(x)$
- $T\varphi(c)$
- $\otimes$

图 30

- $F[(\forall x)(P(x)\rightarrow Q(x))\rightarrow((\forall x)P(x)\rightarrow(\forall x)Q(x))]$
- $T(\forall x)(P(x)\rightarrow Q(x))$
- $F((\forall x)P(x)\rightarrow(\forall x)Q(x))$
- $T(\forall x)P(x)$
- $F(\forall x)Q(x)$
- $FQ(c)$ 一个“新的” c
- $T(\forall x)P(x)$
- $TP(c)$
- $T(\forall x)(P(x)\rightarrow Q(x))$
- $T(P(c)\rightarrow Q(c))$
  - $FP(c)$
    - $\otimes$
  - $TQ(c)$
    - $\otimes$

图 31

- $F((\forall x)(\varphi(x)\wedge\psi(x))\leftrightarrow((\forall x)\varphi(x)\wedge(\forall x)\psi(x)))$
  - $T((\forall x)(\varphi(x)\wedge\psi(x)))$
    - $F((\forall x)\varphi(x)\wedge(\forall x)\psi(x))$
      - $F(\forall x)\varphi(x)$
        - 新的 c $F\varphi(c)$
        - $T((\forall x)(\varphi(x)\wedge\psi(x)))$
        - $T(\varphi(x)\wedge\psi(c))$
        - $T\varphi(c)$
        - $T\psi(c)$
        - $\otimes$
      - $F(\forall x)\psi(x)$
        - $F\psi(d)$ 新的 d
        - $T((\forall x)(\varphi(x)\wedge\psi(x)))$
        - $T(\varphi(d)\wedge\psi(d))$
        - $T\varphi(d)$
        - $T\psi(d)$
        - $\otimes$
  - $F((\forall x)(\varphi(x)\wedge\psi(x)))$
    - $T((\forall x)\varphi(x)\wedge(\forall x)\psi(x))$
    - $T(\forall x)\varphi(x)$
    - $T(\forall x)\psi(x)$
    - $F(\varphi(e)\wedge\psi(e))$ 新的 e
      - $F\varphi(e)$
        - $T(\forall x)\varphi(x)$
        - $T\varphi(e)$
        - $\otimes$
      - $F\psi(e)$
        - $T(\forall x)\psi(x)$
        - $T\psi(e)$
        - $\otimes$

图 32

$T((\forall x)\varphi(x))$和$F((\exists x)\varphi(x))$的原子表说明，对任何基本项t，都能宣布$\varphi(t)$分别为真或为假。而$T((\exists x)\varphi(x))$的原子表说明，只有对某个尚未在表中出现过的常元，才能宣布$\varphi(t)$为真。下面的例子表明：如果不遵守上述规定，就会引来麻烦。

例 6.6 将例 6.3 的蕴涵关系颠倒过来，得到语句$((\exists x)\varphi(x)\to(\forall x)\varphi(x))$，它并非永真。然而，如果违背了上面提到的规则，我们就能生成该语句的一个“证明”，如图 33 所示。

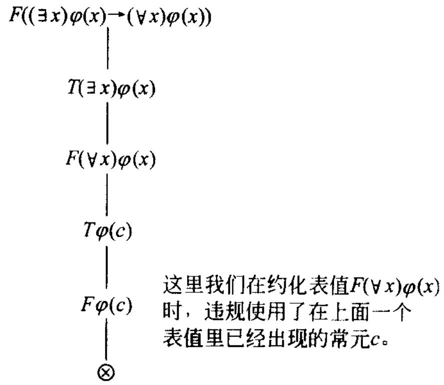


图 33

容易看出，如果没有矛盾出现，谓词逻辑中的表生成永远不会终止。因此这就引发了一个问题：何时才能说一个表值已约化，何时才能说一个表已完成。为了澄清这些定义，首先考虑带有量词的原子表并且考虑如何在构造表时利用它们。在处理$T((\exists x)\varphi(x))$(或$F((\forall x)\varphi(x))$)时，对它作如下分解：对某个尚未出现在分解路径上的常元c，列出$T\varphi(c)$(或$F\varphi(c)$)。原来的语句$(\exists x)\varphi(x)$并不比$\varphi(c)$包含的信息多，因此，此时宣布原子表已完成是合理的。另一方面，在处理$T((\forall x)\varphi(x))$(或$F((\exists x)\varphi(x))$)时情况就完全不同了。此时对任何基本项t，都能添加$T\varphi(t)$(或$F\varphi(t)$)到表中，但这远不足以穷举原来语句所包含的信息。它仅仅提供了$T((\forall x)\varphi(x))$断言下的全称事实的一个例子。因此，不能说原子表$T((\forall x)\varphi(x))$已完成。认清了这一区别，就能定义“表值已约化，表是完成的”等概念。和命题逻辑的情形一样，我们的目标是描述一个系统的程序来生成给定语句φ的一个(从S出发的)表证明。如果φ永真(即φ是S的一个逻辑后承)，则该系统程序总能成功。这就是完全性定理(定理 7.7)要表述的内容。

令$t_1,\cdots,t_n,\cdots$是由语言$\mathcal{L}_C$中所有基本项组成的一个列表，这里$\mathcal{L}_C$包含新的常元c_i。

定义 6.7 令$\tau=\cup\tau_n$为一个(从S出发的)表，P是τ中的一条路径，E是P上的一个表值，并且w是E在P上的第i个出现(即P上第i个标签为E的节点)。

(i)w在P上**已约化**，如果

(1)E不具备$T((\forall x)\varphi(x))$和$F((\exists x)\varphi(x))$两种形式，并且对某个j，τ_{j+1}是由τ_j出发，对E和τ_j上P的一段初始路径应用定义 6.1 中的规则(ii)得到的。(在此情形下我们说，E作为一个原子表的根表值出现在P上。)

或者

(2)E形如$T((\forall x)\varphi(x))$或$F((\exists x)\varphi(x))$，$T\varphi(t_i)$或$F\varphi(t_i)$分别为P上的一个表值，且P

上存在 E 的第 $i+1$ 次出现。

(ii) τ 是**完成的**，如果 τ 上所有表值的每一次出现在包含它的所有非矛盾路径上都已约化（并且对 S 中的每一个 φ，$T\varphi$ 出现在 τ 的所有非矛盾路径上）。否则 τ 就是**未完成的**。

这里的意思是说，涉及带有量词的标号语句，比如 $T(\forall x)\varphi(x)$，只有对我们语言中的每一个项 t_i 实例化才算是完成。现在可以证明：以任意给定的表值为根节点，通过构造适当的（从 S 出发的）完全系统表，总可以得到一个（从 S 出发的）完成表。我们的计划是通过有序的步骤一步步约化每一个表值，从而生成完成表。下面借助节点上的字典序来实现这一计划。

定义 6.8 假设树 T 每层上的叶节点按照左右序排列。我们（由第一章第一节）回忆，如果 T 是一棵二元序列树，那么左右序就是通常的字典序。在 T 的节点 ν，μ 上定义**层字典序**（level-lexicographic ordering）$\leq_{LL}$ 如下：

$\nu\leq_{LL}\mu$，当且仅当 T 中 ν 的层数低于 μ 的层数，或者 ν 和 μ 在同一层而 ν 在 μ 的左边。

定义 6.9 对任意给定的标号语句，我们以其为根节点标签，归纳构造 CST，即**完全系统表**。

(i) 从以标号语句为根的原子表 τ_0 开始。该原子表是唯一确定的，如果在 7a 和 8b 情形下使用项 t_1，而在 7b 和 8a 情形下使用 c_i，i 是符合条件的最小数。

根据归纳，在第 n 步有一个表 τ_n，它可以扩展到 τ_{n+1}。由于 τ_n 是一棵（有穷标签）二叉树，因此可以如上所述为其节点定义层字典序。如果 T 中所有表值的每一次出现都已约化，就终止构造过程。否则，对于 τ_m 中某条非矛盾路径 P 上的某个表值 E 的未约化出现，令 w 为 τ_n 中包含上述未约化出现的节点当中按照层字典序排列最小的那个节点。现在根据下列情形之一进行归纳：

(ii) 如果 E 不具备形式 $T(\forall x)\varphi(x)$ 或 $F(\exists x)\varphi(x)$，那么将以 E 为顶点的原子表连接到 τ 中每一条包含 w 的非矛盾路径的末端。如果 E 形如 $T(\exists x)\varphi(x)$ 或 $F(\forall x)\varphi(x)$，那么使用尚未出现在表中的最小常元 c_j。

(iii) 如果 E 形如 $T(\forall x)\varphi(x)$ 或 $F(\exists x)\varphi(x)$，并且 w 是 E 在 P 上的第 i 个出现，那么分别将

$$\begin{array}{c} E \\ | \\ T\varphi(t_i) \end{array} \qquad \text{或} \qquad \begin{array}{c} E \\ | \\ F\varphi(t_i) \end{array}$$

连接到 τ 中每一条包含 w 的非矛盾路径的末端。

对于带有给定根节点的从前件集合 S 出发的 CST，其定义与上面普通 CST 的定义相似，不同之处在于前者引入了 S 中的元素。在偶数步（$n=2k$），按照上述的 (i)、(ii)、(iii) 三条来生成表。在奇数步（$n=2k+1$），将 $T\alpha_k$ 连接到 τ_n 中的每一条非矛盾路径，从而得到 τ_{n+1}。这里 α_k 是指 S 中的第 k 个元素。我们一直继续从 S 出发的 CST 的构造，直至 S 中所有元素都已放到每一条非矛盾路径之上，并且所有表值的每一次出现在包含它的所有路径上都已约化。

注意，一般来说 CST 是无穷表（即使 S 有穷）。不过重要的是，CST 总是完成表。

命题 6.10 所有 CST 都是完成的。

证明 给定 CSTτ，$\tau_k\subseteq\tau$，考虑 τ_k 中表值 E 的出现在 τ 中非矛盾路径 P 上的任意未约化出现 w。（如果不存在这样的 w，则由定义知 τ 是完成的。）假设 T 中有 n 个节点在层字典

序中小于 w。由 CST 的定义知，在构造 τ_{k+n+1} 时必须约化 P 上的 w。因此，τ 中所有非矛盾路径上每一个表值的每一次出现都已约化，得证。

对于从 S 出发的 CST，可以应用相同的思路来证明：每一条路径上的每一个表值都已约化。(为达此目的所花费的步数是普通 CST 情形的两倍。)在第 $2k+1$ 步添加 S 中的第 k 个元素，这一步骤保证了 S 中的每一个元素都会被放到从 S 出发的 CST 的每一条路径上。由此得到了一个从 S 出发的完成表。 □

例 6.11 图 34 给出了表中只有一个未约化表值的例子。习题 15 问到了这个未约化表值。

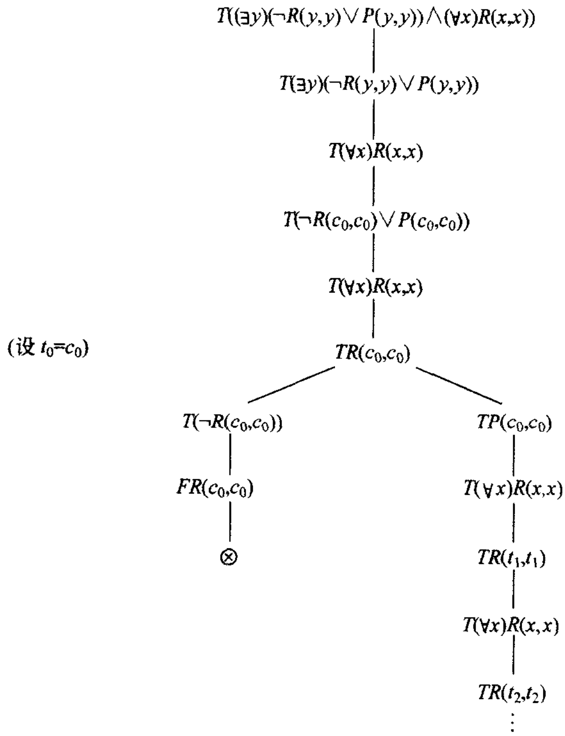


图 34

习题

在习题 1 ~ 11 中，令公式 φ 和 ψ 要么不含自由变元，要么仅含 x 这一个自由变元。给出下列公式的表证明：

1. $(\exists x)(\varphi(x)\vee\psi(x))\leftrightarrow(\exists x)\varphi(x)\vee(\exists x)\psi(x)$。

2. $(\forall x)(\varphi(x)\wedge\psi(x))\leftrightarrow(\forall x)\varphi(x)\wedge(\forall x)\psi(x)$。

3. $(\varphi\vee(\forall x)\psi(x))\rightarrow(\forall x)(\varphi\vee\psi(x))$，$x$ 在 φ 中非自由。

4. $(\varphi\wedge(\exists x)\psi(x))\rightarrow(\exists x)(\varphi\wedge\psi(x))$，$x$ 在 φ 中非自由。

5. $(\exists x)(\varphi\rightarrow\psi(x))\rightarrow(\varphi\rightarrow(\exists x)\psi(x))$，$x$ 在 φ 中非自由。

6. $(\exists x)(\varphi\wedge\psi(x))\rightarrow(\varphi\wedge(\exists x)\psi(x))$，$x$ 在 φ 中非自由。

7. $\neg(\exists x)\varphi(x)\rightarrow(\forall x)\neg\varphi(x)$。

8. $(\forall x)\neg\varphi(x)\rightarrow\neg(\exists x)\varphi(x)$。

9. $(\exists x)\neg\varphi(x)\rightarrow\neg(\forall x)\varphi(x)$。

10. $(\exists x)(\varphi(x)\rightarrow\psi)\rightarrow((\forall x)\varphi(x)\rightarrow\psi)$，$x$ 在 ψ 中非自由。

11. $((\exists x)\varphi(x)\to\psi)\to(\forall x)(\varphi(x)\to\psi)$，$x$ 在 ψ 中非自由。

12. 令 φ 和 ψ 是带有自由变元 x，y，z 的任意公式，w 是不出现在 φ 或 ψ 中的任意变元。给出下列公式的表证明：

a) $\forall x\exists y\neg\forall z\,\varphi(x,\ y,\ z)\leftrightarrow\ \forall x\exists y\exists z\ \neg\varphi(x,\ y,\ z)$。

b) $\exists x\forall y(\ \forall z\ \varphi\vee\psi)\leftrightarrow\ \exists x\forall y\forall w(\varphi(z/w)\vee\psi)$。

c) $\forall x\exists y(\varphi\vee\ \exists z\psi)\leftrightarrow\ \forall x\exists y\exists w(\varphi\vee\psi(z/w))$。

d) $\forall x\exists y(\varphi\to\forall z\psi(z))\to\forall x\exists y\forall w(\varphi\to\psi(z/w))$。

13. **常元定理** 令 $\varphi(x_1,\ \cdots,\ x_n)$为语言$\mathcal{L}$中的一个公式，$x_1$，…，$x_n$ 是其中所有的自由变元。又令 c_1，…，c_n 是不在$\mathcal{L}$中的常元符号。证明：$\forall x_1\cdots\forall x_n\varphi(x_1,\ \cdots,\ x_n)$是表可证的，当且仅当 $\varphi(c_1,\ \cdots,\ c_n)$是表可证的。从语法角度证明，如果给定其中一个公式的证明，那么即可构造另一个公式的证明。(不妨假设公式的证明是由 CST 形式给出。)

14. 如果 T 的每一层上的左右序都是良序(即该序的所有子集都有最小元)，那么序 $\leqslant_{LL}$ 是 T 的节点上的一个良序。

15. 在图 34 中寻找一个未约化的表值，然后给出一个与图 34 具有相同根的完成表。

第七节　表证明的可靠性和完全性

现在可以利用完全系统表来证明谓词逻辑和可证明性中的一些基本定理：可靠性、完全性和紧致性。首先研究谓词逻辑中表证明的可靠性。在本节中，$\mathcal{L}$表示谓词演算中的一个固定语言，S 表示$\mathcal{L}$中的一个语句集合。普通表的情形，即不带前件集合 S，仅是 $S=\varnothing$的情形。由于该注释对本节的所有结论都有效，所以只需处理从 S 出发的表和证明。我们所提到的表的根节点也取自$\mathcal{L}$。

引理 7.1 *令 $\tau=\bigcup\tau_n$ 为一个以 $F\alpha$ 为根从语句集合 S 出发的表。对任何$\mathcal{L}$结构$\mathcal{A}$，如果它是 $S\cup\{\neg\alpha\}$ 的模型，那么它可以扩展成一个与 τ 中某条路径 P 上的所有表值相一致的结构。(注意，$\mathcal{A}$与 $T\alpha(F\alpha)$相一致，如果 α 在$\mathcal{A}$中为真(假)。)*

证明 对出现在 τ 中的所有语句而言，要将$\mathcal{A}$扩展为它们的一个结构，唯一要做的就是为$\mathcal{L}_C-\mathcal{L}$中出现在 P 上的常元 c_i 定义 $c_i^{\mathcal{A}}$。(注意，这些常元在实例化时被当作“新的”常元使用。)

我们通过对序列 τ_n 作归纳来定义 P 和 $c_i^{\mathcal{A}}$，其中序列 τ_n 给出了 τ 的构造。在每一步 n，都有 τ_n 中的一条路径 P_n 以及$\mathcal{A}$的一个扩张$\mathcal{A}_n$(定义域不变)，使得$\mathcal{A}_n$解释 P_n 上所有的 c_i 而且$\mathcal{A}_n$与 P_n 相一致。这足以证明本引理。如果 τ_{n+1}是由 τ_n 通过扩展 P_n 之外的其他路径得到的，那么只需要保持 P_n 或$\mathcal{A}_n$不变。假设 τ_{n+1}是在 P_n 的末端添加一个以 P_n 上某根表值为 E 的原子表或者添加 S 中的一个元素 α_k 得到的。对于后一种情形，只能通过连接 α_k 到 P_n 的末端来扩展 P_n。此时$\mathcal{A}_n$不必扩展，根据假设，它与 α_k(因而与 P_{n+1})相一致。接下来，考虑通过添加以 E 为根的原子表 τ'来扩展 τ_n 的情形。由归纳，假设$\mathcal{A}_n$与 E 相一致。我们希望将$\mathcal{A}_n$扩展到$\mathcal{A}_{n+1}$并寻找 P_n 在 τ_{n+1}中的一条扩展路径 P_{n+1}，使其与$\mathcal{A}_{n+1}$相一致。(这个归纳的基本情形是原子表 τ_0，假设其根节点 $F\alpha$ 与$\mathcal{A}$相一致。对于基本情形的分析与归纳步的情形完全相同：将$\mathcal{A}$扩展到$\mathcal{A}_0$并寻找 τ_0 中的一条路径 P_0，使其与$\mathcal{A}_0$相一致。)我们考虑原子表 τ' 的每一种类型。

(i)命题联结词的情形与命题逻辑可靠性(第一章引理 5.4)的证明完全相同。特别地，$\mathcal{A}_n$不必扩展。例如，如果添加

$$\begin{array}{ccc} & T(\alpha\vee\beta) & \\ \diagup & & \diagdown \\ T\alpha & & T\beta \end{array}$$

那么由归纳知$\mathcal{A}_n \vDash \alpha \vee \beta$，因此$\mathcal{A}_n \vDash \alpha$ 或者$\mathcal{A}_n \vDash \beta$。选取其中一个，相应地扩展 P_n。对于其余命题联结词的分析，留作习题 7。

(ii)如果添加

$$\begin{array}{c} T(\forall x)\varphi(x) \\ | \\ T\varphi(t) \end{array} \qquad \text{或} \qquad \begin{array}{c} F(\exists x)\varphi(x) \\ | \\ F\varphi(t) \end{array}$$

仍然容易处理。$\mathcal{A}_n \vDash \forall x\varphi(x)$（或 $A_n \vDash \neg \exists x\varphi(x)$），因此，$\mathcal{A}_n \vDash \varphi(t)$（$\mathcal{A}_n \vDash \neg\varphi(t)$）。（注意，如果 $t^{\mathcal{A}}$ 在归纳过程中尚未定义，那么现在可任意地定义它并保持归纳假设$\mathcal{A}_n \vDash \forall x\varphi(x)$（或 $A_n \vDash \neg\exists x\varphi(x)$）不变。）

(iii)最后，对某个新的常元 c(即尚未出现在 S 或 P_n 的表值中)，如果添加

$$\begin{array}{c} T(\exists x)\varphi(x) \\ | \\ T\varphi(c) \end{array} \qquad \text{或} \qquad \begin{array}{c} F(\forall x)\varphi(x) \\ | \\ F\varphi(c) \end{array}$$

那么必须定义 $c^{\mathcal{A}}$。由归纳知，$\mathcal{A}_n \vDash \exists x\varphi(x)$（$\mathcal{A}_n \vDash \neg\forall x\varphi(x)$），因此，可以选取一个元素 $a \in A(=A_n)$，使得如果通过定义 $c^{\mathcal{A}} = a$ 将$\mathcal{A}_n$扩展到$\mathcal{A}_{n+1}$，那么有$\mathcal{A}_{n+1} \vDash \varphi(c)$（$\mathcal{A}_{n+1} \vDash \neg\varphi(c)$），得证。 □

定理 7.2(可靠性) *如果存在 α 的一个从 S 出发的表证明 τ，那么 $S \vDash \alpha$。*

证明 如果 $S \vDash \alpha$ 不成立，那么存在一个结构$\mathcal{A}$，使得$\mathcal{A} \vDash \neg\alpha$ 且 S 中每一个 α_k 在$\mathcal{A}$中都为真。引理 7.1 说明，存在 τ 中的一条路径 P 以及$\mathcal{A}$的扩张$\mathcal{A}'$，使得$\mathcal{A}'$与 P 上的所有节点相一致。根据假设，P 是矛盾路径，由此导出了矛盾。 □

现在开始研究谓词逻辑中表证明方法的完全性。和命题逻辑的情形(第一章定理 5.3，尤其是引理 5.4)一样，计划利用 CST 中的一条非矛盾路径来构造$\mathcal{L}_C$的一个结构，使其与 P 上的所有表值相一致。这里的基本思想是从唯一可用的素材——语法对象，尤其是出现在路径上的基本项出发，构造要求的结构。这一思想及其在完全性定理证明中的应用对其他许多重要结论的证明有着很强的借鉴意义，这其中包括厄布朗定理(定理 10.4)和斯科朗-勒文海姆定理(定理 7.7)。

定理 7.3 *假设 τ 是以 $F\alpha$ 为根的从 S 出发的一个完全系统表，P 是 τ 中一条非矛盾路径。于是存在一个结构$\mathcal{A}$，使得 α 在其中为假而 S 中的所有语句在其中为真。*

证明 令扩展语言$\mathcal{L}_C$的基本项的主列表(master list)上的那些基本项 t_i 构成的集合 A 为该结构的定义域。根据$\mathcal{L}_C$的语法，将$f^{\mathcal{A}}$ 与 n-元函数符号 f 自然地联系起来，从而定义$f^{\mathcal{A}}$ 如下：

$$f^{\mathcal{A}}(t_{i_1}, t_{i_2} \cdots, t_{i_n}) = f(t_{i_1}, \cdots, t_{i_n})$$

注意，我们结构中的元素都是基本项，因此，出现在上面等式左边的 t_i 可以看作是结构中被函数$f^{\mathcal{A}}$ 作用的元素。等式的右边也是一个项，它同样可以看作是结构中的元素，将其称作函数值。若 R 是一个 n-元谓词字母，则通过路径 P 定义 $R^{\mathcal{A}}$ 如下：

$$R^{\mathcal{A}}(t_{i_1}, \cdots, t_{i_n})，\text{当且仅当 } TR(t_{i_1}, \cdots, t_{i_n}) \text{ 是 } P \text{ 上的一个表值。}$$

下面首先通过归纳法给出一个更强的断言，然后利用该断言推出本定理。 □

引理 7.4 令符号标记如上所示。

(i)若 $F\beta$ 出现在 P 上，则 β 在 $\mathcal{A}$ 中为假。

(ii)若 $T\beta$ 出现在 P 上，则 β 在 $\mathcal{A}$ 中为真。

证明 首先回忆命题 6.10，P 上所有表值的每一次出现在 P 上都已约化。现在对 β 的深度（准确地说是 β 所对应的辅助形成树的深度，见定义 3.8）作归纳。

(i)若 β 是一个原子语句，则 β 形如 $R(t_{i_1},\cdots,t_{i_n})$。若 $T\beta$ 出现在 P 上，则 $R^{\mathcal{A}}$ 关于 $t_{i_1},\cdots,t_{i_n}$ 为真。若 $F\beta$ 出现在 P 上，则由于 P 是非矛盾的，于是 $T\beta$ 不出现在 P 上且 $R^{\mathcal{A}}$ 关于 $t_{i_1},\cdots,t_{i_n}$ 为假。

(ii)假设 β 是通过联结词构造得来，例如，β 是 $(\beta_1\vee\beta_2)$。由于 τ 是完成表，因此，如果 $T\beta$ 出现在 P 上，那么 $T\beta_1$，$T\beta_2$ 之一出现在 P 上。根据归纳假设，如果 $T\beta_1$ 出现在 P 上，那么 β_1 在 $\mathcal{A}$ 中为真（β_2 的情况与此相似）。因此，β_1，β_2 之一为真，于是 $(\beta_1\vee\beta_2)$ 在 $\mathcal{A}$ 中为真（根据真值的归纳定义）。另一方面，如果 $F(\beta_1\vee\beta_2)$ 出现在 P 上，那么 $F\beta_1$ 和 $F\beta_2$ 同时出现在 P 上。由归纳假设知，β_1 和 β_2 在 $\mathcal{A}$ 中均为假，于是 $(\beta_1\vee\beta_2)$ 在 $\mathcal{A}$ 中为假。其余联结词的情形与此相似，留作习题 8。

(iii)假设 β 形如 $(\forall v)\varphi(v)$。如果 w 是 $T((\forall v)\varphi(v))$ 在 P 上的第 i 次出现，那么 $T\varphi(t_i)$ 出现在 P 上且 P 上存在 $T((\forall x)\varphi(x))$ 的第 $i+1$ 次出现。因此，如果 $T((\forall v)\varphi(v))$ 出现在 P 上，那么对每一个基本项 t，$\varphi(t)$ 出现在 P 上。由于 $\varphi(t)$ 的深度低于 $(\forall v)\varphi(v)$，由归纳假设知，对每一个基本项 t，$\varphi(t)$ 在 $\mathcal{A}$ 中为真。由于这些项构成了结构 $\mathcal{A}$ 的域，故 $(\forall v)\varphi(v)$ 在 $\mathcal{A}$ 中为真。

如果 $F(\forall v)\varphi(v)$ 出现在 P 上，那么由于 τ 是完成表，所以对某个 t，$F\varphi(t)$ 出现在 P 上。由归纳假设知 $\varphi(t)$ 在 $\mathcal{A}$ 中为假，因此 $(\forall v)\varphi(v)$ 在 $\mathcal{A}$ 中为假。

(iv) $\exists v\varphi(v)$ 的情形与 $(\forall v)\varphi(v)$ 相似，留作习题 9。 □

至此完成了定理 7.3 的证明。 □

现在将关于证明有穷性的通用注释应用到完全系统表上来。

命题 7.5 如果一个完全系统表的所有路径都是矛盾的，那么它是一个有穷表。

证明 根据构造过程，如果某条路径上出现矛盾，便不再扩展该路径。因此，CST 上的所有矛盾路径都是有穷的。由库尼西引理（第一章定理 1.4），本命题得证。 □

事实上现在已经证明了完全性定理的一个能行版本。对任意给定的语句 α 以及任意语句集合 S，要么可以生成一个证明：α 是 S 的一个逻辑后承，要么可以生成 S 的一个模型，α 在其中失败。

推论 7.6 对 $\mathcal{L}$ 中的每一个语句 α 以及每一个语句集合 S，要么

(i)以 $F\alpha$ 为根的从 S 出发的 CST 是 α 的一个从 S 出发的表证明

要么

(ii)完全系统表中存在一个非矛盾分支，它生成一个结构，使得 α 在其中为假而 S 的所有元素在其中为真。

由于推论 7.6(ii)中的路径是可数的（即该路径的符号（因而它的项和公式）与自然数的一个子集之间存在一一对应），因此与之相应的结构也是可数的。实际上我们已经证明了斯科朗-勒文海姆定理。

定理 7.7(斯科朗-勒文海姆(Skolem-Löwenheim)) 如果一个可数的语句集合 S 可满足(即有一个模型),那么它有一个可数的模型。

证明 考虑以矛盾式 $\alpha \wedge \neg \alpha$ 为根的从 S 出发的 CST。根据可靠性定理(定理 7.2),它不可能是 $\alpha \wedge \neg \alpha$ 的一个从 S 出发的表证明。因此,它必有一条非矛盾路径 P。由于$\mathcal{L}_C$中只有可数多个基本项,因此定理 7.4 的证明中所定义的结构即是我们要求的 S 的可数模型。 □

对任意序数也有类似的结论。同时注意,我们所讲的可数是指至多可数,因此,模型也可能是有穷的。不过在我们的假设条件下,模型总是无穷的(见习题 3)。我们在第三章第五节会提到等号的一种特殊应用,在此情形下语句集合只有有穷模型。如果忽略有关 PROLOG 的注释,读者可以直接跳到该节了解这种特殊情形。

与命题逻辑中的完全性和可靠性定理类似,我们也可以从可证明性与逻辑后承的等价关系出发,重新表述推论 7.6。需要注意的是,若 α 在 S 的某个模型中为假,则其不可能是 S 的一个逻辑后承。

定理 7.8(完全性和可靠性)

(i)α 是从 S 出发表可证的,当且仅当 α 是 S 的一个逻辑后承。

(ii)如果取 α 为任意矛盾式,例如(i)中的 $\beta \wedge \neg \beta$,那么 S 是不相容的,当且仅当 S 不可满足。

谓词逻辑的紧致性定理也是上述结果的一个推论。

定理 7.9(紧致性) 令 $S=\{\alpha_1, \alpha_2, \cdots\}$ 为谓词逻辑中的一个语句集合。S 可满足,当且仅当 S 的所有有穷子集可满足。

证明 必要性容易证明。对于充分性,考虑带有根表值 $F(\alpha \wedge \neg \alpha)$ 的从 S 出发的 CST。如果该 CST 是矛盾的,由命题 7.5 知它是有穷的。如果它是无穷的,那么它有一条非矛盾路径,根据推论 7.6,存在一个结构,使得所有 α_i 在其中为真。如果它是矛盾的并且有穷,那么 $\alpha \wedge \neg \alpha$ 就是 S 的某个有穷子集的一个逻辑后承,该有穷子集的元素都出现在表中。由于 $\alpha \wedge \neg \alpha$ 没有模型,因此,该有穷子集也没有模型。 □

我们应当指出谓词逻辑与命题逻辑中完全性证明的一个主要差别。命题逻辑中的完成表总是有穷的,因此,对每一个命题 α 我们都能能行地判定它是否永真或者生成一个反例。对于谓词逻辑,如果给定的语句 φ 是永真的,我们最终能够找到一个证明。另一方面,如果它不是永真的,那么完成表以及提供反例的路径都将是无穷的。因此,在这样的构造中我们永远无法知道 φ 不是永真的。这一现象是不可避免的。丘奇定理告诉我们,没有能行的方法能够判定一个给定的语句是否在谓词演算中永真。作为 PROLOG 程序终止的相关结论的一个推论,我们将在第三章推论 8.10 中证明。然而,基于第三章第八节习题 3 所表述的语义方法,现在即可证明这一推论。

习题

1. 给出习题 6.13(常元定理)的一个语义证明:令 $\varphi(x_1, \cdots, x_n)$ 是语言$\mathcal{L}$中的公式,所有自由变元都列了出来。令 $c_1, \cdots, c_n$ 是不在$\mathcal{L}$中的常元符号。证明:语句 $\forall x_1 \cdots \forall x_n \varphi(x_1, \cdots, x_n)$ 是表可证的,当且仅当 $\varphi(c_1, \cdots, c_n)$ 是表可证的。(提示:首先证明 $\varphi(x_1, \cdots, x_n)$ 永真,当且仅当 $\varphi(c_1, \cdots, c_n)$ 永真。然后再应用完全性定理。)
2. 寻找一个有穷语言$\mathcal{L}$以及$\mathcal{L}$中一个有穷的语句集合 S,使得 S 有一个论域无穷的模型,但没有论域有穷的模型。

3. 令$\mathcal{L}$是谓词逻辑中的语言，S是$\mathcal{L}$中语句的集合。证明：S可满足，当且仅当它有一个无穷模型。
4. 令$\mathcal{L}$是自然数集$\mathcal{N}$上的算术语言，且0，1，+，·，>包含其中。$\mathcal{N}(=\{0,1,2,\cdots\})$。令$\mathrm{Th}(\mathcal{N})$是$\mathcal{L}$中所有在$\mathcal{N}$上为真的那些语句的集合。证明：存在$\mathrm{Th}(\mathcal{N})$的一个非标准模型，即存在$\mathcal{L}$上的一个结构$\mathcal{M}$，使得$\mathrm{Th}(\mathcal{N})$的所有语句在其中为真，然而其中存在一个元素$c$大于所有的$n\in\mathcal{N}$。
5. 再次考虑命题逻辑习题中紧致性的应用。利用谓词逻辑给出第一章第六节习题7，习题8的简化证明。（注意，图G在谓词逻辑中可以表示成平面图，因为根据库拉托夫斯基(Kuratowski)的一个定理，该结论等价于某两个指定的不是G的子图的有穷图。）
6. **演绎定理** 令Σ是语言$\mathcal{L}$上一个有穷的语句集合，$\wedge\Sigma$是该集合所有元素的合取。证明：对$\mathcal{L}$上的任何语句φ，下列各式等价：
   (i) $\Sigma\vDash\varphi$
   (ii) $\vDash\wedge\Sigma\to\varphi$
   (iii) $\Sigma\vdash\varphi$
   (iv) $\vdash\wedge\Sigma\to\varphi$
7. 通过描述P_n在其他联结词下的扩张，完成引理7.1中情形(i)的证明。
8. 通过处理其他联结词，完成引理7.4中情形(ii)的证明。
9. 通过考虑$\beta=\exists v\varphi(v)$的情形，完成引理7.4中情形(iv)的证明。
10. 令$\mathcal{L}$是不含函数符号的语言。任给一个形如$\forall x_1\cdots\forall x_n\exists y_1\cdots\exists y_m\varphi$（$\varphi$中不含量词）的语句$\psi$，判定$\psi$是否永真。（提示：首先利用第六节习题13将$\psi$的永真性归约成$\exists y_1\cdots\exists y_m\varphi(c_1,\cdots,c_n,y_1,\cdots,y_m)$的永真性，其中$c_1,\cdots,c_n$是新的常元。如果问题仍然困难，那么等到第十节习题6再考虑它。）
11. 令R是二元关系符号，$R^{\mathcal{A}}$是R在结构$\mathcal{A}$下的解释。$R^{\mathcal{A}}$的传递闭包(transitive closure)是指所有序对(a,b)组成的集合，并且对任意的(a,b)，存在从a到b的一条有穷$R^{\mathcal{A}}$路径，即$\mathcal{A}$中元素构成的一个序列$a_0,a_1,\cdots,a_n(n\geqslant 1)$，其中$a_0=a$，$a_n=b$且$\mathcal{R}^{\mathcal{A}}(a_i,a_{i+1})$，$0\leqslant i<n$。证明：传递闭包不是一阶可定义的，即证明：不存在谓词逻辑中的公式$\mathrm{TC}(x,y)$，使得对所有结构$\mathcal{A}$以及$a,b\in\mathcal{A}$，$\mathcal{A}\vDash\mathrm{TC}(a,b)$，当且仅当$(a,b)$在$R^{\mathcal{A}}$的传递闭包之中。（提示：归纳定义公式$\rho_n(x,y)$如下，
$$\rho_1(x,y)\equiv R(x,y)$$
$$\rho_{n+1}(x,y)\equiv\exists z(R(x,z)\wedge\rho_n(z,y))。$$
证明，在任何结构$\mathcal{A}$中，序对(a,b)在$R^{\mathcal{A}}$的传递闭包之中，当且仅当对某个n，有$\mathcal{A}\vDash\rho_n(a,b)$。假设存在这样一个公式$\mathrm{TC}(x,y)$来表示$R$的传递闭包。考虑下述无穷的语句集合
$$\{\mathrm{TC}(a,b)\}\cup\{\neg\rho_n(a,b)\mid n\geqslant 1\}。$$
利用谓词逻辑的紧致性导出矛盾。）

*第八节 公理化方法

和对命题逻辑一样，我们通过公理和规则简要介绍一种研究谓词逻辑的经典方法。为简洁起见，像第一章第七节那样仅用$\neg$和$\to$两个命题联结词。同理我们也把存在量词$\exists$看作已被定义的符号：即用$\neg\forall x\neg\varphi(x)$来替换$\exists x\varphi(x)$。（由第四节习题5知它们等价。）此外固定一些常元、函数符号以及谓词符号来构造语言$\mathcal{L}$。这里所研究的公理仍包括命题逻辑（第一章公理7.1）中的那些模式，不过现在变元α，β，γ可以是$\mathcal{L}$中的任何公式。在此基础上我们增加了两条模式来表达全称量词的含义。注意，现在考虑的是所有公式而不只是语句，含有自由变元的公式的永真性与其全称闭包的永真性相同。

公理8.1 令α，β和γ分别为$\mathcal{L}$中的任意公式。系统中的公理为如下公式：

(i) $(\alpha\to(\beta\to\alpha))$。

(ii) $((\alpha\to(\beta\to\gamma))\to((\alpha\to\beta)\to(\alpha\to\gamma)))$。

(iii) $((\neg\alpha)\to(\alpha\to\beta))$。

(iv) $(\forall x)\alpha(x)\to\alpha(t)$，对任意项 t，t 在 α 中对 x 可替换。

(v) $(\forall x)(\alpha\to\beta)\to(\alpha\to(\forall x)\beta)$，如果 α 中不含 x 的自由出现。

容易验证以上公理模式的所有实例都是永真的。正如我们在定义可替换性(定义 2.8)时解释的那样，(iv)中的限制是必要的。回顾在例 2.9(ii)中考虑的结构$\mathcal{Z}$：带有常元0 和1 的整数、后继函数 s 以及意指 $x+y=z$ 的谓词 $A(x, y, z)$。特别地，考虑真语句 $\varphi=\forall x\exists yA(x, y, 0)$。作为一个真的全称语句，$\varphi$ 应该对任何个体为真。事实上(iv)断言任何合理替换都致使公式在结构$\mathcal{Z}$下永真。另一方面，如果用 $s(y)$ 替换 x，那么得到 $\forall x\exists yA(s(y), y, 0)$，它在结构$\mathcal{Z}$下为假。至于(v)中的限制，考虑($\mathcal{Z}$中的)真语句 $\varphi=\forall x(\forall yA(x, y, y)\to A(x, 1, 1))$。如果忽略(v)中的限制，可以从 φ 出发(通过下面的演绎规则)推出 $\forall yA(x, y, y)\to\forall xA(x, 1, 1)$。该公式在$\mathcal{Z}$中并非永真，将 x 的自由出现设为 0 即可看出这一点。(这个替换仅仅影响蕴涵式的左边——使之变为真语句 $\forall yA(0, y, y)$，然而蕴涵式的右边为假。)

我们的系统有两条推理规则。第一条是应用于$\mathcal{L}$中公式的假言推理，第二条是证明了下述等价性一个方面的一般化推理：带有自由变元的公式的永真性与其全称闭包的永真性等价。

(该等价性的另一方面由公理模式(iv)推出。用 t 替换 x 即可。)

规则 8.2

(i)**假言推理**：对任何公式 α 和 β，从 α 和 $\alpha\to\beta$ 出发，可以推出 β。

(ii)**一般化推理**(generalization)：从 α 出发推出 $\forall x\alpha$。

和在命题逻辑中一样，一般把这些基于公理与规则的系统叫做希尔伯特式证明系统。对于从公式集合 Σ 出发的证明，其定义与命题逻辑的情形基本相同，只不过现在多了几条公理和规则。

定义 8.3 令 Σ 为$\mathcal{L}$中公式的一个集合。

(i)**从 Σ 出发的证明**是指$\mathcal{L}$中公式的有穷序列 $\alpha_1, \alpha_2, \cdots, \alpha_n$，并且对每个 $i\leqslant n$，下列表述之一为真：

(1) α_i 是 Σ 中的成员。

(2) α_i 是一条公理。

(3) α_i 可由其前的某个 α_j 出发应用推理规则推出。

(ii) α 是**从 Σ 出发可证的**(定理)，如果存在一个从 Σ 出发的证明 $\alpha_1, \cdots, \alpha_n$，使得 $\alpha_n=\alpha$。

(iii) α 的**证明**就是从 $\varnothing$ 出发的证明。α 是可证的，如果它是从 $\varnothing$ 出发**可证的**。

对于这里所述的系统，标准的可靠性、完全性和紧致性定理均可证明。参阅 Elliot Mendelson 的《Introduction to Mathematical Logic》[1979, 3.2]，这本书还发展了谓词逻辑。在第十三节，把消解的基于规则的系统扩展到谓词逻辑，并证明相应的结果。

第九节　前束范式和斯科朗化

我们希望说明，在某种意义下，谓词逻辑几乎可以归约到命题逻辑。粗略地讲，希望通过引入新的函数符号和项来消去量词。基本思想如下：把公式

$$\varphi=\forall x_1\cdots\forall x_n\exists y_1\cdots\exists y_mR(x_1,\cdots,x_n,y_1,\cdots,y_m)$$

替换成

$$\psi = \forall x_1 \cdots \forall x_n R(x_1, \cdots, x_n, f_1(x_1, \cdots, x_n), f_2(x_1, \cdots, x_n), \cdots, f_n(x_1, \cdots, x_n))$$

这里每个f_i都是一个新的函数符号。关于f_i的解释如下：对任意给定的x_1，…，x_n，选择一个使得上述公式为真的y_i，如果存在的话。这样的函数叫做斯科朗函数。显然φ与ψ是可满足等价的(equisatisfiable)(即φ可满足当且仅当ψ可满足)，因此，如果找到了ψ的一个表(或其他)反驳，也就找到了φ的一个反驳。为了充分利用上述过程，首先把φ替换成其前束范式(prenex normal form)φ'，在前束范式中所有量词都出现在开头。然后通过引入适当的斯科朗函数来消去形如$\forall \vec{x} \exists \vec{y}$的量词块。最终目标是得到一个与原公式$\varphi$可满足等价(我们说$\varphi$与$\psi$可满足等价，如果它们同时可满足或者同时不可满足。)的全称公式(universal formula)ψ(即仅含全称量词且全部出现在ψ的开头)。至此我们只需考虑任意反驳证明模式中的那些全称公式。(当然，我们心里要想到消解。)

我们已经知道如何用$\neg$与$\vee$来表示联结词($\neg$与$\vee$构成了联结词的一个充分集，见第一章推论2.11)，下面不妨假设给出的公式φ不含其他联结词。现在来说明如何寻找φ的前束范式。首先需要处理的是$\neg$与$\vee$。

引理9.1 *对任意的量词串$\overrightarrow{Qx} = Q_1x_1Q_2x_2 \cdots Q_nx_n$(每个$Q_i$是$\forall$或者$\exists$)以及任意公式$\varphi$，$\psi$，有下列可证的等价关系：*

(1a)$\vdash \overrightarrow{Qx} \neg \forall y \varphi \leftrightarrow \overrightarrow{Qx} \exists y \neg \varphi$

(1b)$\vdash \overrightarrow{Qx} \neg \exists y \varphi \leftrightarrow \overrightarrow{Qx} \forall y \neg \varphi$

(2a)$\vdash \overrightarrow{Qx}(\forall y \varphi \vee \psi) \leftrightarrow \overrightarrow{Qx} \forall z(\varphi(y/z) \vee \psi)$

(2a′)$\vdash \overrightarrow{Qx}(\varphi \vee \forall y \psi) \leftrightarrow \overrightarrow{Qx} \forall z(\varphi \vee \psi(y/z))$

(2b)$\vdash \overrightarrow{Qx}(\exists y \varphi \vee \psi) \leftrightarrow \overrightarrow{Qx} \exists z(\varphi(y/z) \vee \psi)$

(2b′)$\vdash \overrightarrow{Qx}(\varphi \vee \exists y \psi) \leftrightarrow \overrightarrow{Qx} \exists z(\varphi \vee \psi(y/z))$。

这里变元z不出现在φ和ψ中，且z不在x_i中。

证明 这些等价关系的表证明相当简单，留作习题。((1a)，(2a)，(2b′)的证明分别为第六节习题12(a)，(b)，(c)。)另外也可以先从语义角度考虑等价性，然后再应用完全性定理。(第四节习题5实际上给出了(1a)和(1b)。)习题1～3将会给出这些等价关系证明的一般方法。 □

备注 在消解证明中，像(2a)和(2b)那样重新命名变元以避免可能的冲突，这种办法通常叫做标准化变元分离。

现在可以证明所有公式φ都有前束范式。

定理9.2(前束范式) *对每一个公式φ，存在一个含有相同自由变元的等价公式φ'，在其中所有的量词都出现在开头。φ的这个等价式叫做φ的***前束范式***(PNF)。*

证明 对φ的深度作归纳。注意，由第一章推论2.11知，可以假设φ中仅含$\neg$和$\vee$两个联结词。如果φ是原子公式，则不证自明。如果φ是$\forall y \psi$或$\exists y \psi$，ψ'是ψ的PNF，那么$\forall y \psi'$或$\exists y \psi'$是φ的PNF。(这是习题1中归纳的基本情形)。如果$\varphi = \neg \psi$，且ψ'是ψ的PNF，那么反复应用引理的(1a)和(1b)即可生成φ的PNF。如果$\varphi = \psi \vee \theta$，那么反复应用引理的(2a)，(2a′)，(2b)和(2b′)即可生成φ的PNF。 □

备注 我们也可以引入相关的前束规则来直接处理其他联结词。利用下列等价关系把公式转化成PNF时，我们只需处理$\leftrightarrow$这一个联结词：

(3a)$\vdash \overrightarrow{Qx}(\forall y\varphi \wedge \psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi(y/z) \wedge \psi)$

(3a')$\vdash \overrightarrow{Qx}(\varphi \wedge \forall y\psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi \wedge \psi(y/z))$

(3b)$\vdash \overrightarrow{Qx}(\exists y\varphi \wedge \psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi(y/z) \wedge \psi)$

(3b')$\vdash \overrightarrow{Qx}(\varphi \wedge \exists y\psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi \wedge \psi(y/z))$

(4a)$\vdash \overrightarrow{Qx}(\forall y\varphi \to \psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi(y/z) \to \psi)$

(4a')$\vdash \overrightarrow{Qx}(\varphi \to \forall y\psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi \to \psi(y/z))$

(4b)$\vdash \overrightarrow{Qx}(\exists y\varphi \to \psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi(y/z) \to \psi)$

(4b')$\vdash \overrightarrow{Qx}(\varphi \to \exists y\psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi \to \psi(y/z))$。

变元 z 不出现在各个等价关系的左边。

例 9.3 寻找下面两个公式的 PNF：

(i) $\forall x \exists y P(x, y) \vee \neg \exists x \forall y Q(x, y)$：

$\forall u[\exists y P(u, y) \vee \neg \exists x \forall y Q(x, y)]$

$\forall u \exists v[P(u, v) \vee \neg \exists x \forall y Q(x, y)]$

$\forall u \exists v[P(u, v) \vee \forall x \neg \forall y Q(x, y)]$

$\forall u \exists v[P(u, v) \vee \forall x \exists y \neg Q(x, y)]$

$\forall u \exists v \forall w[P(u, v) \vee \exists y \neg Q(w, y)]$

$\forall u \exists v \forall w \exists z[P(u, v) \vee \neg Q(w, z)]$。

(ii) $\forall x \forall y[(\exists z)(P(x, z) \wedge P(y, z)) \to \exists u Q(x, y, u)]$：

$\forall x \forall y \forall w[P(x, w) \wedge P(y, w) \to \exists u Q(x, y, u)]$

$\forall x \forall y \forall w \exists z[P(x, w) \wedge P(y, w) \to Q(x, y, z)]$。

(iii)通过如下过程可以得到公式(i)的另一个 PNF：

$\forall u[\exists y P(u, y) \vee \neg \exists x \forall y Q(x, y)]$

$\forall u[\exists y P(u, y) \vee \forall x \neg \forall y Q(x, y)]$

$\forall u \forall w[\exists y P(u, y) \vee \neg \forall y Q(w, y)]$

$\forall u \forall w \exists v[P(u, v) \vee \neg \forall y Q(w, y)]$

$\forall u \forall w \exists v[P(u, v) \vee \exists y \neg Q(w, y)]$

$\forall u \forall w \exists v \exists z[P(u, v) \vee \neg Q(w, z)]$。

现在可以把谓词演算中任意语句的反驳证明问题归约成该语句全称公式的反驳证明问题。

定理 9.4(斯科朗化) *对给定语言 $\mathcal{L}$ 中的所有语句 φ，在通过添加新的函数符号得到的扩展语言 $\mathcal{L}'$ 中，存在一个全称公式 φ'，使得 φ 与 φ' 是可满足等价的。*

(注意，我们没有断言公式是等价的。该过程总能生成一个 φ'，使得 $\varphi' \to \varphi$ 永真，但 $\varphi \to \varphi'$ 不必永真。第九节习题 4 给出了一个例子。)

证明 根据定理 9.2，假设 φ 已是前束范式。令 $y_1, \cdots, y_n$ 是 φ 中存在量词约束的变元，它们在 φ 中从左往右依次出现。对每个 $i \leqslant n$，令 $x_1, \cdots, x_{n_i}$ 为全称量词约束的变元，且其出现在 y_i 之前。对每个 $i \leqslant n$，添加新的 n_i-元函数符号 f_i，从而把语言 $\mathcal{L}$ 扩展成 $\mathcal{L}'$。现在通过如下过程生成 φ'：首先删去每个 $\exists y_i$，然后用 $f_i(x_1, \cdots, x_{n_i})$ 替换 y_i 的每个剩余出现。我们断言 φ' 即是要找的与 φ 可满足等价的语句。验证该断言需要多次用到下面的引理。 □

引理 9.5 对语言$\mathcal{L}$中的任意语句 $\varphi=\forall x_1\cdots\forall x_n\exists y\psi$，$\varphi$ 与 $\varphi'=\forall x_1\cdots\forall x_n\psi(y/f(x_1,\cdots,x_n))$是可满足等价的，如果函数符号$f$不在$\mathcal{L}$中。

证明 令$\mathcal{L}'$是通过对$\mathcal{L}$添加函数符号f得到的新语言。显然，如果$\mathcal{A}'$是$\mathcal{L}'$的一个结构，$\mathcal{A}$是$\mathcal{A}'$省去函数符号f对应的解释而得到的结构，且$\mathcal{A}'\vDash\varphi'$，那么$\mathcal{A}\vDash\varphi$。另一方面，如果$\mathcal{A}$是$\mathcal{L}$的一个结构，且$\mathcal{A}\vDash\varphi$，我们可以把结构$\mathcal{A}$扩展成$\mathcal{A}'$，方法如下：定义$f^{\mathcal{A}'}$，使得对每一个 $a_1,\cdots,a_n\in A=A'$，$\mathcal{A}\vDash\psi(y/f(a_1,\cdots,a_n))$。显然，$\mathcal{A}'\vDash\varphi'$。注意，$n$可能为0，即$f$可能是一个常元符号。 □

推论 9.6 对语言$\mathcal{L}$中语句的任意集合 S，我们可以构造语言$\mathcal{L}'$中全称语句的一个集合S'，使得 S 与 S'是可满足等价的。这里$\mathcal{L}'$是通过对$\mathcal{L}$添加新的函数符号得到的扩展语言。

证明 对 S 中的每个语句 φ，分别应用定理 9.4 提供的构造方法引入新的函数符号f_φ，并生成相应的全称语句 φ'。令 S'是所有这些语句 φ'的集合，$\mathcal{L}'$是$\mathcal{L}$对应的扩展。和定理 9.4 的证明一样，易知，如果$\mathcal{L}'$的某结构$\mathcal{A}'$是 S'的一个模型，那么它也是 S 的模型。如何将 S 的模型扩展成 S'的模型，定理 9.4 的证明给出了方法：每次定义一个新的函数符号f_φ，使其与已经定义的函数符号相互独立。 □

例 9.7 与例 9.3 中前束范式相对应的斯科朗化结果如下：

(i) $\forall u\forall w[P(u,f_1(u))\vee\neg Q(w,f_2(u,w))]$。

(ii) $\forall x\forall y\forall w[P(x,w)\wedge P(y,w)\rightarrow Q(x,y,f(x,y,w))]$。

(iii) $\forall u\forall w[P(u,f_1(u,w))\vee\neg Q(w,f_2(u,w))]$。

例 9.8 对于标准数学结构(比如群和环)的公理系统，它们的构造过程中就有很多常见的斯科朗化的例子。通过引入适当的斯科朗函数，形如 $\forall x\exists y\varphi(x,y)$的公理可以替换成开公式 $\varphi(x,f(x))$。

作为一个特例，我们再来考虑例 2.9 中的结构：整数$\mathcal{Z}$，语句 $\forall x\exists yA(x,y,0)$表示每个整数都有一个相反数。该语句的斯科朗化结果为 $\forall xA(x,f(x),0)$。f是指把每个整数 x 映射到其相反数 $-x$ 的单元函数，于是斯科朗化的语句即是指，对所有 x，$x+(-x)=0$。

从第五节介绍的谓词逻辑的子句形式来看，每一个语句集合都有一个与之可满足等价的子句形式。

推论 9.9 对$\mathcal{L}$的任意语句集合 S，存在(采用第五节中的术语)一个公式，即语言$\mathcal{L}'$中的子句集合 T，使得 S 和 T 是可满足等价的。这里$\mathcal{L}'$是对$\mathcal{L}$添加新的函数符号得到的。

证明 考虑推论 9.6 给出的由全称语句 $\forall\vec{x}\varphi'(\vec{x})$构成的集合 S'，它与 S 可满足等价。将 S'中元素省去开头的全称量词，得到的开公式 $\varphi'(\vec{x})$的集合记作 T'。(由第四节习题 8 或第六节习题 13 知，φ 与 φ'等价。)如果将$\mathcal{L}'$中的每个原子公式视作一个命题字母，且对每个公式 $\varphi'\in T'$，记其 CNF 等价式为 $\psi_\varphi=\wedge\psi_{\varphi,i}$，那么得到一个 CNF 公式的集合 T''，其各公式与 T'中公式对应等价：对每个 $\varphi\in S$，$\wedge\psi_{\varphi,i}=\psi_\varphi\equiv\varphi'\equiv\varphi$。(由定理 4.8 知，对每个 φ，ψ_φ等价于 φ'。)于是，要找的子句集合 T 就由基于 T''中公式 φ 的所有合取式构成：$T=\{\psi_{\varphi,i}\mid\varphi\in S\}$。 □

习题

1. 令 φ 和 ψ 为任意公式(带有自由变元)，令$\overrightarrow{Qx}$表示任意的量词串 $Q_1x_1Q_2x_2\cdots Q_nx_n$。证明：如果 φ 和 ψ 等价，则$\overrightarrow{Qx}\varphi$ 与$\overrightarrow{Qx}\psi$ 等价。(提示：对$\overrightarrow{Qx}$的长度 n 作归纳。)因此，在证明(1a)～(4b')时，假设那些公式带有自由变元，而初始的量词串$\overrightarrow{Qx}$为空。

2. 利用常元定理(第四节习题 8)说明：在(1a)～(4b′)中，同样可以假设公式中不带自由变元。
3. 从语义或者表证明角度讨论(1a)～(4b′)的永真性。(利用习题 1 和 2，假设$\overrightarrow{Qx}$为空，且公式中不带自由变元。)
4. 令 $\varphi(x, y)$为原子公式，f为不出现在 φ 中的函数符号。证明：语句 $\forall x\varphi(x, f(x)) \to \forall x \exists y\varphi(x, y)$ 永真，但它的逆 $\forall x \exists y\varphi(x, y) \to \forall x\varphi(x, f(x))$ 非永真。
5. 找出下列语句的前束范式并斯科朗化：
   (a) $\forall y(\exists xP(x, y) \to Q(y, z)) \wedge \exists y(\forall xR(x, y) \vee Q(x, y))$。
   (b) $\exists xR(x, y) \leftrightarrow \forall yP(x, y)$。
   (c) $\forall x \exists yQ(x, y) \vee \exists x \forall yP(x, y) \wedge \neg \exists x \exists yP(x, y)$。
   (d) $\neg(\forall x \exists yP(x, y) \to \exists x \exists yR(x, y)) \wedge \forall x \neg \exists yQ(x, y)$。

第十节 厄布朗定理

对于不可满足性的二分法以及表证明的完全性定理所蕴涵的模型构造，从语句集合到全称公式集合的归约以及斯科朗函数的引入为它们提供了更具体的方法。对已经包含各种斯科朗函数的语言$\mathcal{L}$，考虑其中任意全称语句构成的集合 S。假设$\mathcal{L}$中至少包含一个常元 c。我们断言，要么 S 是不相容的(即不可满足的)，要么存在 S 的一个模型$\mathcal{A}$，其元素是语言$\mathcal{L}$的基本项。由于在$\mathcal{L}$的每一个结构中这些基本项都有相应的解释，因此，在某种意义上可以说，这就是$\mathcal{L}$的一个极小结构。

定义 10.1 语言$\mathcal{L}$中基本(变元自由)项的集合叫做$\mathcal{L}$的**厄布朗域**(Herbrand universe)。$\mathcal{L}$的结构$\mathcal{A}$叫做**厄布朗结构**(Herbrand structure)，如果$\mathcal{A}$的域 A 是$\mathcal{L}$的厄布朗域，且对$\mathcal{L}$中每一个函数符号 f 以及 A 中元素 $t_1, \cdots, t_n$，

$$f^{\mathcal{A}}(t_1, \cdots, t_n) = f(t_1, \cdots, t_n)$$

(这里要求对$\mathcal{L}$中的每个常元符号 c，$c^{\mathcal{A}} = c$。)

厄布朗域让我们联想到完全性定理(定理 7.3)证明中生成的结构。正如我们所见，它们的确密切相关。同时注意，我们对$\mathcal{L}$中谓词的解释没有限制，因此对给定的语言$\mathcal{L}$可以有很多厄布朗结构。

定义 10.2 如果 S 是语言$\mathcal{L}$语句的一个集合，那么 S 的**厄布朗模型**(Herbrand model)$\mathcal{M}$是 S 的模型，且是$\mathcal{L}$的厄布朗结构，即 S 的所有语句在$\mathcal{M}$中为真。

例 10.3 如果语言$\mathcal{L}$包含常元 a，c，一元函数符号 f，二元函数符号 g，谓词 P，Q，R，那么$\mathcal{L}$的厄布朗域 H 为：

$$\{a, c, f(a), f(c), g(a,c), ff(a), ff(c), f(g(a,c)), g(a, f(a)), g(a, f(c)), \\ \cdots, g(a, g(a,c)), \cdots, g(f(a), f(c)), \cdots, fff(a), \cdots\}$$

断言：对由全称语句(或开公式)组成的任何相容集合 S，存在一个厄布朗模型。如果 S 是不相容的，则其不可满足性取决于基于公式基本实例(即用厄布朗结构中的项替换 S 中由全称量词约束的那些(自由)变元得到的实例)的真值函数。

定理 10.4(厄布朗定理) 令 $S = \{\varphi_i(x_1, \cdots, x_{n_i})\}$ 为语言$\mathcal{L}$中开公式的一个集合。以下两条结论之一成立：

(i) S 有一个厄布朗模型。

(ii) S 是不可满足的，特别地，存在 S 中元素的有穷多个基本实例，其合取式是不可满足的。

后一条结论等价于

(ii′)存在 S 中公式的否定的有穷多个基本实例，其析取式永真。(由于我们可以将这些基本实例看作从命题字母出发构造而来的，所以析取式永真等价于它是一个真值函数重言式。)

证明 令 S'是由从$\mathcal{L}$出发的 S 中公式的所有基本实例组成。对任意语句 α，考虑由 $F(\alpha\wedge\neg\alpha)$开始从 S'(仅在语言$\mathcal{L}$中，即不添加任何常元符号)出发的 CST。此时有两种可能的结果。其一，该表中有一条(可能无穷的)非矛盾路径。在此情形下，定理 7.3 的证明给出了 S'的一个模型$\mathcal{A}$，$\mathcal{A}$中元素就是$\mathcal{L}$中的基本项，即$\mathcal{A}$是 S'的一个厄布朗模型。根据 S'的定义以及从 S'出发表证明的定义，对每一个 $\varphi\in S$ 以及厄布朗域中每一组 t_1，…，t_n，$\varphi(t_1,\cdots,t_n)$在$\mathcal{A}$中为真。因此，由路径定义出来的厄布朗域上的结构$\mathcal{A}$是 S 的一个模型。

其二，该表有穷且矛盾。在此情形下，根据定义，对 S'的出现在该表中的那些元素构成的集合，该表是其不可满足性的一个证明。因此，得到如(ii)所要求的不可满足的合取式。此外，S 不可满足：在 S 的任意模型中，对每一个 $\varphi_i\in S$，$\varphi_i(x_1,\cdots,x_{n_i})$永真，即 $\varphi_i(x_1,\cdots,x_{n_i})$对自由变元 x_1，…，x_{n_i}的每一个实例为真。然而(ii)中的任何例子都能直接给出上述实例的一个集合，使得该集合中的例子在任何模型中都不能同时被满足。

最后，根据定理 4.8，将变元自由的公式视作命题字母。于是根据命题规则知，(ii)中所要求的合取式的不可满足性等价于各成员之否定的析取式的永真性或重言式。因此，(ii)与(ii′)等价。 □

注意，如果 S 不可满足(因而(i)不成立)，那么(ii)直接给出了 S 的不可满足性。因此，我们要么能够生成 S 的一个厄布朗模型，要么能够生成一个特殊的有穷反例来说明 S 没有模型。

现在对厄布朗定理作一些变化，这对进一步研究消解定理证明以及 PROLOG 将会特别有用。我们也可以应用该结果，将谓词逻辑的可证明性或永真性直接归约到命题逻辑上来。下面首先考虑一个特殊的存在公式：

推论 10.5 如果 $\varphi(\vec{x})$是语言$\mathcal{L}$中的一个无量词约束的公式，且至少含一个常元符号，那么 $\exists\vec{x}\varphi(\vec{x})$永真，当且仅当存在$\mathcal{L}$的基本项$\vec{t}_i$，使得 $\varphi(\vec{t}_1)\vee\cdots\vee\varphi(\vec{t}_n)$是重言式。

证明 首先注意，$\exists\vec{x}\varphi(\vec{x})$永真 $\Leftrightarrow$ $\forall\vec{x}\neg\varphi(\vec{x})$不可满足 $\Leftrightarrow$ $\neg\varphi(\vec{x})$不可满足。根据定理 10.4(ii)，$\neg\varphi(\vec{x})$不可满足，当且仅当存在$\mathcal{L}$的有穷多的基本项$\vec{t}_i$，使得 $\varphi(\vec{t}_1)\vee\cdots\vee\varphi(\vec{t}_n)$是重言式。 □

把这些结果转化成第五节的子句术语，就能得到下面的结论，它将对谓词演算中的消解定理证明起到关键作用。

定理 10.6 子句集合 S 不可满足，当且仅当 S 中子句的厄布朗域上的所有基本实例的集合 S'不可满足。

证明 如果 S 中元素的一些实例(厄布朗域中的项构成的实例)组成的某个集合不可满足，则维护其成员子句永真性的公式 S 肯定不可满足。对于充分性，如果 S 不可满足，则由厄布朗定理(ii)知，存在 S 中子句的实例组成的一个有穷集合，它是不可满足的。 □

对于前面的厄布朗定理，作如下限制：S 仅包含全称公式。(或者等价地，只考虑子句

的集合。)习题1中的例子说明我们所作的限制是必要的。另一方面，如果进一步限制 S 仅包含程序子句，则可以证实极小(事实上是最小)厄布朗模型的存在。(见习题3。)此外，考虑PROLOG感兴趣的一种情形：对一个从某个程序子句集合出发的演绎，我们可以像推论10.5那样仅使用一个单独的永真实例，从而消去析取式，也即是说，如果 P 是一个程序子句集合，$\theta(\vec{x})$ 是一个原子公式，那么 $P \vDash \exists \vec{x}\theta(\vec{x}) \Leftrightarrow$ 存在厄布朗项 $\vec{t}$，使得 $P \vDash \theta(\vec{t})$(习题5)。

最后，尽管与消解定理证明没有直接关联，我们还是可以通过斯科朗化将推论10.5推广到任意语句的情形。该结果给出了谓词逻辑中的永真性在命题逻辑中的等价形式。

定理 10.7 令 φ 为语言 $\mathcal{L}$ 中一个前束范式语句，ψ 为等价于 $\neg\varphi$ 的前束范式，$\theta(\vec{x})$ 为 ψ 在语言 $\mathcal{L}'$ 中的开斯科朗化，$\mathcal{L}'$ 如定理9.4所示。(注意，ψ 中的自由变元恰好是 φ 中的那些被存在量词约束的变元。)则 φ 永真，当且仅当存在 $\mathcal{L}'$ 中项 $\vec{t}_1, \cdots, \vec{t}_n$，使得 $\neg\theta(\vec{t}_1) \vee \cdots \vee \neg\theta(\vec{t}_n)$ 是重言式。

证明 根据推论10.5，需要证明 φ 永真，当且仅当 $\exists \vec{x} \neg\theta(\vec{x})$ 永真。现在我们知道 φ 永真，当且仅当 $\neg\varphi$ 不可满足。另一方面，定理9.4说明 $\neg\varphi$ 可满足，当且仅当 $\theta(\vec{x})$ 可满足。因此，φ 永真，当且仅当 $\theta(\vec{x})$ 不可满足。最后注意，$\theta(\vec{x})$(或者等价地，$\forall \vec{x}\theta$)不可满足，当且仅当 $\exists \vec{x} \neg\theta(\vec{x})$ 永真。 □

习题

1. 令 $\mathcal{L}$ 由常元 c 和一元谓词 R 组成。
   (a) $\mathcal{L}$ 的厄布朗域是什么?
   (b) $\mathcal{L}$ 可能的厄布朗结构是什么?
   (c) 令 $S = \{R(c), \exists x \neg R(x)\}$。注意，$S$ 不完全由全称公式组成，所以 S 不是子句形式。证明：如果 S 可满足，则其无厄布朗模型。
2. 令 $\mathcal{L}$ 由常元 c、函数 f 和一元谓词 R 组成。
   (a) $\mathcal{L}$ 的厄布朗域是什么?
   (b) 列举出 $\mathcal{L}$ 的无穷多可能的厄布朗结构。
3. 证明：每一个程序子句的集合 P 有一个极小(事实上在集合包含的意义下是最小)厄布朗模型。(提示：证明 P 的所有厄布朗模型的交仍是 P 的厄布朗模型。)
4. 令 M_P 为语言 $\mathcal{L}$ 中程序子句的集合 P 的极小厄布朗模型。证明：对 $\mathcal{L}$ 的每一个原子语句 φ，$M_P \vDash \varphi$，当且仅当 φ 是 P 的一个逻辑后承。
5. 令 P 是程序子句的集合，$G = \neg\theta(\vec{x})$ 是一个目标子句。证明：如果 $P \vDash \exists \vec{x}\theta(\vec{x})$(或者等价地，$P \cup \{G\}$ 是不可满足的)，那么存在厄布朗项 $\vec{t}$，使得 $P \vDash \theta(\vec{t})$。(提示：如果 $P \vDash \exists \vec{x}\theta(\vec{x})$，观察极小模型 $\mathcal{M}_P$ 并应用习题4。)
6. 令 $\mathcal{L}$ 为不含函数符号的语言。描述一个程序，使得对给定的任何形如 $\forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m \varphi$($\varphi$ 不含任何量词)的语句 ψ，判定 ψ 是否永真。上述形式的语句叫做语言 $\mathcal{L}$ 的 $\forall\exists$-语句。(提示：首先应用第六节习题13将 ψ 的永真性归约到 $\exists y_1 \cdots \exists y_m \varphi(c_1, \cdots, c_n, y_1, \cdots, y_m)$ 的永真性，这里 $c_1, \cdots, c_n$ 为新的常元。然后应用推论10.5。)

第十一节　合　一

如我们在定理9.4所见，对谓词逻辑中的每一个公式 φ，存在另外一个开公式 ψ，使得它是合取范式并且其可满足性等价于 φ。因此，如果我们要考虑谓词逻辑中公式(的集合)的可满足性，就要考虑子句形式的开公式。它与命题逻辑的情形唯一的差别在于，这里的文字

是原子公式(可能含有自由变元或新加进来的斯科朗函数符号)而不仅仅是命题字母。当然，含有自由变元的子句与其全称闭包等价。从消解定理证明的观点来看，要从 S 出发演绎出□,谓词逻辑与命题逻辑之间唯一的差别在于，如何在可用子句里将自由变元实例化(即作替换)才能应用归约规则。

当然，我们可以像厄布朗定理的表证明那样，只列出厄布朗结构中的所有基本项替换，并以此为输入运行我们的消解机器。很明显这不是一个高效的程序。我们需要更好的想法。

例如，如果有两个子句 $C_1=\{P(f(x),y),\neg Q(a,b,x)\}$，$C_2=\{\neg P(f(g(c)),g(d))\}$,我们应当能够直接将 x，y 分别替换成 $g(c)$，$g(d)$，从而消解 C_1 与 C_2 得到 $\{\neg Q(a,b,g(c))\}$。(注意 C_1 与其全称闭包 $\forall x\forall y(P(f(x),y)\vee\neg Q(a,b,x))$ 等价，从全称闭包出发可以演绎出任何替换实例。)消解证明中作替换的一般方法叫做合一(unification)(或者匹配(matching))。在给出谓词演算的消解算法之前，首先描述合一法。要描述合一法，我们需要一些与替换相关的概念。

定义 11.1 **替换** θ 是指形如 $\{x_1/t_1,x_2/t_2,\cdots,x_n/t_n\}$ 的有穷集合，这里 x_i 是互不相同的变元，每个 t_i 是一个不同于 x_i 的项。如果 t_i 都是基本项，我们就把 θ 叫做**基本替换**(ground substitution)。如果 t_i 是互不相同的变元，我们就把 θ 叫做**改名替换**(renaming substitution)。

由于要考虑子句中的替换，必须定义 θ 在子句 C 上的操作。为了定义不同替换的结合(连续运用)，先来定义替换 θ 在项上的操作。

定义 11.2 **表达式**(expression)是指项或文字。给定替换 θ 与表达式 E(或者表达式的集合 S)，对所有 $i\leqslant n$，用 t_i 替换 x_i 在 E(S 的所有元素)中的每一次出现，将得到的结果记作 $E\theta$($S\theta$)。如果结果表达式 $E\theta$(表达式的集合 $S\theta$)是基本项，即没有变元，那么该替换叫做 $E(S)$ 的一个**基本实例**。

备注 把替换 θ 写成由形如 x_i/t_i 的一些元素构成的集合，而非这些项的序列。这是因为 t_i 替换 x_i 是同时进行的，没有先后顺序。因此，在 $E\{x_1/t_1,x_2/t_2\}$ 中，t_2 替换 x_2 不会对 t_1 中 x_2 的任何出现造成影响。

例 11.3

(i)令 $S=\{f(x,g(y)),P(a,x),Q(y,z,b),\neg P(y,x)\}$，$\theta=\{x/h(a),y/g(b),z/c\}$。那么 $S\theta=\{f(h(a),g(g(b))),P(a,h(a)),Q(g(b),c,b),\neg P(g(b),h(a))\}$。这里 θ 是一个基本替换，$S\theta$ 是 S 的一个基本实例。

(ii)令 S 如上所述，$\sigma=\{x/h(y),y/g(z),z/c\}$。那么

$$S\sigma=\{f(h(y),g(g(z))),P(a,h(y)),Q(g(z),c,b),\neg P(g(z),h(y))\}。$$

替换结合是关于替换的一种自然操作，即定义替换 $\theta\sigma$ 如下：对任意表达式 E 应用替换 $\theta\sigma$ 得到的 $E(\theta\sigma)$，与对 $E\theta$ 应用 σ 得到的 $(E\theta)\sigma$ 完全相同。

例 11.4 令 $E=P(x,y,w,u)$，考虑替换 $\theta=\{x/f(y),y/g(z),w/v\}$ 与 $\sigma=\{x/a,y/b,z/f(y),v/w,u/c\}$。那么 $E\theta=P(f(y),g(z),v,u)$，$(E\theta)\sigma=P(f(b),g(f(y)),w,c)$。$\theta\sigma$ 是什么呢？首先用 $f(y)$ 替换 x，然后用 b 替换 y，得到 $x/f(b)$。接下来用 $g(z)$ 替换 y，然后用 $f(y)$ 替换 z，如此得到 $y/g(f(y))$。先用 v 替换 w，再用 w 替换 v，其实就是 w/w，它不会引起任何变换，故而可以略去。对 σ 的替换 x/a 同样不影响最终结果，因为在应

用 θ 之后 σ 中已经没有 x。最后，对 σ 的替换 u/c 可以顺利进行，因为对 u 的任何替换都与 θ 无关。因此，$\theta\sigma=\{x/f(b),\ y/g(f(y)),\ u/c,\ z/f(y),\ v/w\}$。

根据上面这个例子，我们可以给出替换结合的形式定义。

定义 11.5

(i) 如果 $\theta=\{x_1/t_1,\ \cdots,\ x_n/t_n\}$，$\sigma=\{y_1/s_1,\ \cdots,\ y_m/s_m\}$，那么 $\theta\sigma$ 就是替换 $\{x_1/t_1\sigma,\ \cdots,\ x_n/t_n\sigma,\ y_1/s_1,\ \cdots,\ y_m/s_m\}$，其中要删除 $x_i=t_i\sigma$ 所对应的替换 $x_i/t_i\sigma$ 以及 $y_j\in\{x_1,\ \cdots,\ x_n\}$ 的那些替换 y_j/s_j。

(ii) **空替换**(empty substitution) ϵ(对任何表达式都不做任何变动)是恒等操作，即 $\theta\epsilon=\epsilon\theta=\theta$ 对所有替换 θ 都成立。

现在验证上面所定义的替换结合是正确的并且满足结合律。

命题 11.6 对任意表达式 E 以及替换 $\theta,\ \psi,\ \sigma$：

(i) $(E\theta)\sigma=E(\theta\sigma)$。

(ii) $(\psi\theta)\sigma=\psi(\theta\sigma)$。

证明 令 θ 与 σ 如替换结合的定义所述，$\psi=\{z_1/r_1,\ \cdots,\ z_k/r_k\}$。由于替换只是将表达式中的每个变元替换成某个项，所以在(i)中需要考虑 E 为变元的情形，比如 $E=v$。在(ii)中对 v 分别应用 $(\psi\theta)\sigma$，$\psi(\theta\sigma)$，验证结果是否相同。

(i) 我们分两种情形进行讨论：

情形 1：$v\notin\{x_1,\ \cdots,\ x_n\}$。在此情形下 $v\theta=v$，$(v\theta)\sigma=v\sigma$。如果 $v\notin\{y_1,\ \cdots,\ y_m\}$，那么 $v\notin\{x_1,\ \cdots,\ x_n,\ y_1,\ \cdots,\ y_m\}$，从而 $v\sigma=v=v(\theta\sigma)$，因此 v 没有任何变动。另一方面，如果对某个 $j\leqslant n$，$v=y_j$，那么 $y_j\notin\{x_1,\ \cdots,\ x_n\}$，$(v\theta)\sigma=v\sigma=s_j=v(\theta\sigma)$。

情形 2：对某个 $i\leqslant n$，$v=x_i$。在此情形下 $v\theta=t_i$，$(v\theta)\sigma=t_i\sigma$，而根据定义，这恰好是 $v(\theta\sigma)$。

(ii) 反复应用(i)可得：

$$\begin{aligned} v((\psi\theta)\sigma) &= (v(\psi\theta))\sigma \\ &= ((v\psi)\theta)\sigma \\ &= (v\psi)(\theta\sigma) \\ &= v(\psi(\theta\sigma))。\end{aligned}$$

□

因此，在书写多个替换相结合时可以略去括号，但替换上的结合操作不是可交换的。(习题 3 要求读者寻找一个反例。)

对于替换，我们最感兴趣的是将不同子句里的某些元素化为相同表达式以便我们应用消解规则。这个替换过程叫做合一。

定义 11.7 令 $S=\{E_1,\ \cdots,\ E_n\}$ 为一个表达式集合，我们说替换 θ 是 S 的**合一子**(unifier)，如果 $E_1\theta=E_2\theta=\cdots=E_n\theta$，即 $S\theta$ 是单元素集。S 是**可合一的**(unifiable)，如果它有一个合一子。

例 11.8

(i) $\{P(x,\ a),\ P(b,\ c)\}$ 与 $\{P(f(x),\ z),\ P(a,\ w)\}$ 都是不可合一的(见习题 2)。

(ii) $S_1=\{P(x,\ c),\ P(b,\ c)\}$ 与 $S_2=\{P(f(x),\ y),\ P(f(a),\ w)\}$ 都是可合一的。S_1 可由 $\{x/b\}$ 合一，并且仅需这一个替换。S_2 的情形就不大相同了。$\theta=\{x/a,\ y/w\}$ 可以合一 S_2，但是 $\sigma=\{x/a,\ y/a,\ w/a\}$，$\psi=\{x/a,\ y/b,\ w/b\}$ 以及其他很多替换同样能够合一 S_2。

这里，与其他替换相比，θ 的优势在于它给以后的替换提供了更多选择。如果首先应用 θ 来合一 S_2，那么得到的表达式 $P(f(a), c)$ 可以应用 $\{w/c\}$ 来合一。如果首先应用 σ 或 ψ，下面就无法进行。另一方面，可以对 θ 分别应用替换 $\{w/a\}$，$\{w/b\}$ 得到 σ，ψ。我们在下面的定义中表述了 θ 的这个性质。

定义 11.9 S 的合一子 θ 是它的一个**最一般合一子**(most general unifier, mgu)，如果对 S 的任意同一子 σ，存在替换 λ，使得 $\theta\lambda=\sigma$。

在重命名变元时应用 mgu，结果是唯一的：

定理 11.10 如果 θ，ψ 都是 S 的 mgu，那么存在改名替换 σ 与 λ(改名替换只是将一些互不相同的变元替换成另外一些互不相同的变元)，使得 $S\theta\sigma=S\psi$ 且 $S\theta=S\psi\lambda$。

证明 由 mgu 的定义知，存在 σ 与 λ，使得 $S\theta\sigma=S\psi$ 且 $S\psi\lambda=S\theta$。不妨设 σ 与 λ 分别只对出现在 $S\theta$ 与 $S\psi$ 中的变元作替换。(由于 θ 与 ψ 都能合一 S，所以 $S\theta$ 与 $S\psi$ 分别由单个项 $E\theta$ 与 $E\psi$ 组成。)假设 σ 作替换 x_i/t_i，这里 t_i 不是变元或常元。在此情形下，$S\theta\sigma=\{E\theta\sigma\}$ 中表达式 $E\theta\sigma$ 的复杂度(比如长度)严格大于 $S\theta$ 中 $E\theta$ 的复杂度。由于用项来替换变元(比如 λ)不能缩短表达式的长度，因此我们无法得到要求的结果：$S\psi\lambda=S\theta\sigma\lambda=S\theta$。如果 σ 中有替换 x_i/c，c 是某个常元，那么没有替换(比如 λ)能够使 $S\theta\sigma$ 中表达式 $E\theta\sigma$ 的 c 回到 $E\theta\in S\theta$ 中变元 x_i 的实例。因此对任何 λ，都无法得到 $S\theta\sigma\lambda=S\theta$。现在我们知道 σ 可以包含如下替换：仅将一个变元替换成另外一个变元。如果 σ 对互不相同的一些变元应用上述替换，那么 λ 就不再能够区分它们了。因此 σ(同样地 λ)仅是改名替换。 □

习题

1. 寻找替换来合一下列表达式的集合：
   (a) $\{P(x, f(y), z), P(g(a), f(w), u), P(v, f(b), c)\}$。
   (b) $\{Q(h(x, y), w), Q(h(g(v), a), f(v)), Q(h(g(v), a), f(b))\}$。
2. 解释为什么例 8(i) 中的表达式都不可合一。
3. 证明：替换的结合不是可交换的，即寻找两个替换 σ，λ，使得 $\sigma\lambda\neq\lambda\sigma$。
4. 我们说表达式 E 和 F 互为彼此的变形(variant)(或者 E 是 F 的另一种形式)，如果存在替换 θ，ψ，使得 $E\theta=F$ 且 $F\psi=E$。注意改名替换是指形如 $\{x_1/y_1, \cdots, x_n/y_n\}$ 的替换，这里 $x_1, \cdots, x_n$ 是 E 中互不相同的变元，$y_1, \cdots, y_n$ 是 F 中互不相同的变元，并且对任何 $i\leq n$，$y_i\neq x_i$。证明：如果 E 和 F 互为彼此的变形，那么存在 E 的一个改名替换 σ，使得 $E\sigma=F$。

第十二节　合一算法

本节给出一个算法来寻找表达式的有穷集合 S 的一个 mgu。先来看两个例子，$S_1=\{f(x, g(x)), f(h(y), g(h(z)))\}$，$S_2=\{f(h(x), g(x)), f(g(x), h(x))\}$。分别寻找 S_1 和 S_2 的 mgu，注意两种情形中要合一的项都是以 f 为开始。显然，如果是以不同的函数或谓词符号为开始，那么下面就无法进行，因为合一操作仅仅替换变元。下一步就是验证$S_1(S_2)$ 各项中 f 的第一项和第二项。如果可以用一个替换同时合一它们，那么就可以合一 $S_1(S_2)$。对 S_1，分别有 $T_1=\{x, h(y)\}$，$T_2=\{g(x), g(h(z))\}$。为了合一 T_1，最通用的办法就是用 $h(y)$ 替换 x。此时 T_2 也必须作相应的替换，T_2 中两项分别变为 $g(h(y))$，$g(h(z))$。这两项的第一处差别出现在 y 处。现在可以应用 $\{y/z\}$ 合一它们。注意，所有替换都必须应用到整个表达式当中。至此，分别得到 $f(h(z), g(h(z)))$，$f(h(z), g(h(z)))$。因此，

$\{x/h(y)\}\{y/z\}=\{x/h(z),\ y/z\}$即是我们要求的合一子。对 S_2，当我们尝试合一 f 的变元时，寻找合一子的过程就终止了。这里，对于 S_2 各项中 f 的第一项$\{h(x),\ g(x)\}$，函数符号而非变元出现了差别，因而没有希望合一它们，S_2 是不可合一的。

通常的合一算法是沿着给定集合中的每个表达式，确定第一处出现不一致符号的位置。如果在所有表达式中该位置都不是变元，则给定的集合不可合一。如果某个表达式在该位置上出现变元，则可以用另外一个表达式中相对应的项来替换它。只要被替换的变元不出现在要替换它的项当中，该替换就向着合一的方向前进了。反复执行这一过程，直至表达式集合被合一。下面我们形式化表述该过程。

定义 12.1 令 S 是一个有穷的非空表达式集合。为了定义 S 的**不一致集合**(disagreement set)，寻找 S 中所有元素 E 出现不完全相同符号的第一处(即最左边)位置。对每个 $E\in S$，由此位置开始的那些子表达式的集合叫做 S 的不一致集合，记作 $D(S)$。(从形成树的角度看，是寻找每个表达式所对应的形成树上按字典序排列最小的那个节点，使得所有这些节点的标签不是由相同的符号开始。$D(S)$就是这些节点标签的集合。)

注意，S 的任何合一子 ψ 都必须能够合一 $D(S)$。

例 12.2 对前面考虑过的表达式集合 $S_1=\{f(x,\ g(x)),\ f(h(y),\ g(h(z)))\}$和 $S_2=\{f(h(x),g(x)),\ f(g(x),\ h(x))\}$，不一致集合分别为 $D(S_1)=\{x,\ h(y)\}$，$D(S_2)=\{h(x),g(x)\}$。对 $T_1=S_1\{x/h(y)\}=\{f(h(y),\ g(h(y))),\ f(h(y),\ g(h(z)))\}$，不一致集合为$\{y,\ z\}$。

算法 12.3(S 的合一算法) 令 S 是一个表达式集合。合一 S 的过程如下：

第 0 步：令 $S_0=S$，$\sigma_0=\epsilon$。

第 $k+1$ 步：如果 S_k 是单元素集合，终止算法并宣布 $\sigma_0\sigma_1\sigma_2\cdots\sigma_k$ 即是 S 的一个 mgu。若不然，观察是否存在变元 v 以及不包含 v 的项 t，使得两者都在 $D(S_k)$ 中。如果不存在，终止算法并宣布 S 没有 mgu。(注意，在此情形下我们至少清楚 S_k 是不可合一的。)如果存在，令 v 和 t 是满足条件的(在项的任何固定序中)最小的。(事实上，可以非确定地选取这样的 v 和 t 让算法继续进行，这在本节最后一个定理证明中会有更清楚的阐述。)令 $\sigma_{k+1}=\{v/t\}$，$S_{k+1}=S_k\sigma_{k+1}$，然后跳到第 $k+2$ 步。

例 12.4 考虑表达式的集合

$$S=\{P(f(y,g(z)),h(b)),P(f(h(w),g(a)),t),P(f(h(b),g(z)),y)\}$$

第 1 步：$S=S\epsilon=S_0\sigma_0$ 不是单元素集合。$D(S_0)=\{y,\ h(w),\ h(b)\}$。根据项的不同排序，$\sigma_1$ 有两种可能：$\{y/h(w)\}$和$\{y/h(b)\}$。事实上选择后者更优(见第 2 步)，但是假设我们并不知道，仍令 $\sigma_1=\{y/h(w)\}$。则得到 $S_1=S_0\sigma_1$ 如下：

$$\{P(f(h(w),g(z)),h(b)),P(f(h(w),g(a)),t),P(f(h(b),g(z)),h(w))\}$$

第 2 步：$D(S_1)=\{w,\ b\}$，$\sigma_2=\{w/b\}$(结合第 1 步，我们其实作了替换$\{y/h(b)\}$)。则 S_2 是

$$\{P(f(h(b),g(z)),h(b)),P(f(h(b),g(a)),t),P(f(h(b),g(z)),h(b))\}$$

第 3 步：$D(S_2)=\{z,\ a\}$，$\sigma_3=\{z/a\}$。则 S_3 是

$$\{P(f(h(b),g(a)),h(b)),P(f(h(b),g(a)),t),P(f(h(b),g(a)),h(b))\}$$

第 4 步：$D(S_3)=\{h(b),\ t\}$，$\sigma_4=\{t/h(b)\}$。则 S_4 是

$$\{P(f(h(b),g(a)),h(b)),P(f(h(b),g(a)),h(b)),P(f(h(b),g(a)),h(b))\}$$

第 5 步：S_4 是单元素集合，于是 S 的 mgu 是

$$\{y/h(w)\}\{w/b\}\{z/a\}\{t/h(b)\} = \{y/h(b), w/b, z/a, t/h(b)\}$$

定理 12.5 对任何 S，合一算法在某 $k+1$ 步终止，且给出一个正确解，即要么宣布 S 是不可合一的，要么给出 S 的一个 mgu，即 $\psi=\sigma_0\sigma_1\cdots\sigma_k$。此外，$\psi$ 有一个特殊性质，即对 S 的任意合一子 θ，$\theta=\psi\theta$。

证明 首先，算法总会终止，因为 S 中变元的数目有穷，而算法的每一非终止步都会消除 S 中某个变元的所有出现。显然，如果算法终止时宣布不存在合一子，那么 S 就是不可合一的。另一方面，如果算法终止时宣布 $\psi=\sigma_0\cdots\sigma_n$ 是 S 的一个 mgu，那么至少 ψ 是 S 的一个合一子。假设 θ 是 S 的任意合一子，我们断言 $\theta=\psi\theta$。下面归纳证明对每一个 i，$\theta=\sigma_0\cdots\sigma_i\theta$。

对 $i=0$，断言显然成立。假设 $\theta=\sigma_0\sigma_1\cdots\sigma_i\theta$ 且 $\sigma_{i+1}=\{v/t\}$。现在只要证明替换 $\sigma_{i+1}\theta$ 与 θ 相同。为此证明它们对所有变元的操作都是相同的。对 $x\neq v$，$x\sigma_{i+1}\theta$ 显然与 $x\theta$ 相同。对 v 本身，$v\sigma_{i+1}\theta=t\theta$。由于 θ 合一 $S\sigma_0\cdots\sigma_i$，并且 v 和 t 属于 $D(S\sigma_0\cdots\sigma_i)$，因此 θ 也必然合一 v 和 t，即 $t\theta=v\theta$。 □

这里给出的合一算法简单但是低效。搜索 v 和 t，使得 v 不出现于 t 中这一过程可能会花费大量时间，因为我们很可能要检查不一致集合中的每一对条目，而不是简单地取我们碰到的第一个变元和项。例如，考虑合一 $S=\{P(x_1,\ \cdots,\ x_n),\ P(f(x_0,\ x_0),\ \cdots,\ f(x_{n-1},\ x_{n-1}))\}$：

$$D(S_0) = \{x_1, f(x_0,x_0)\};\sigma_1 = \{x_1/f(x_0,x_0)\};$$

$$S_1 = \{P(f(x_0,x_0),x_2,\cdots,x_n),P(f(x_0,x_0),$$
$$f(f(x_0,x_0),f(x_0,x_0)),f(x_2,x_2),\cdots,f(x_{n-1},x_{n-1}))\}。$$

$$D(S_1) = \{x_2, f(f(x_0,x_0),f(x_0,x_0))\};\sigma_2 = \{x_2/f(f(x_0,x_0),f(x_0,x_0))\};\text{等等}$$

注意，在宣布 σ_1 之前必须验证，x_1 不是 $f(x_0,\ x_0)$ 中变元的两个出现中的任何一个。对 σ_2 有四个出现要验证。一般来说，$D(S_{i+1})$ 中变元的出现数目是 $D(S_i)$ 的两倍，因此“验证出现”需要花费指数时间。

事实上，现在已有一些更为高效的（甚至是线性时间的）合一算法（见 Martelli，Montannari [1982，5.4]）。不幸的是，当前所有的 PROLOG 执行程序都只是简单地略去“验证出现”的过程。它们仅仅选取 $D(S_k)$ 中第一个变元 x，并在贡献 x 到 $D(S_k)$ 中的那些表达式中选取第一个不同于 x 的项 t 来替换 x。因此，执行程序相信 $S=\{x,\ f(x)\}$ 是可合一的。（实际上它们无法进行替换操作。它们尝试用 $f(x)$ 替换 x，然后返回 x，之后再用 $f(x)$ 替换 x，如此反复进行。）显然，这种合一方式损坏了消解方法的可靠性。因此，我们考虑将一些针对这种不正确演绎的保护措施添加到程序当中。当然，只有对 PROLOG 的演绎机制有了更充分的认识，我们才能讨论这一问题，见第三章第二节。不幸的是，程序员所能做的几乎所有努力都无法完全补偿略去“验证出现”所带来的损失。不过在本章推论 8.7 证明了，有一些程序在不验证出现的情况下仍然能够正确运行，并且这些程序足以演算所有能行函数。

习题

1. 对下列集合应用合一算法，找出其 mgu 或者说明其 mgu 不存在。

(a) $\{P(x,\ y),\ P(y,\ f(z))\}$

(b) $\{P(a, y, f(y)), P(z, z, u)\}$

(c) $\{P(x, g(x)), P(y, y)\}$

(d) $\{P(x, g(x), y), P(z, u, g(a)), P(a, g(a), v)\}$

(e) $\{P(g(x), y), P(y, y), P(y, f(u))\}$。

2. 对下列集合应用合一算法，找出它们的 mgu 或者说明它们是不可合一的。

(a) $\{P(h(y), a, z), P(h(f(w)), a, w), P(h(f(a)), a, u)\}$

(b) $\{P(h(y), a, z), P(h(f(w)), a, w), P(h(f(a)), a, b)\}$。

第十三节 消 解

现在描述如何将合一算法和命题逻辑中的消解方法结合起来，从而给出消解方法在完全谓词逻辑中的证明模式。和前面一样，我们考虑子句形式的公式。值得注意的是，这里的文字是指带有适当自由变元的原子公式或其否定。第九节和第十节的结果表明，只要添加函数符号到语言当中，每一个语句都会有一个可满足等价的子句形式。注意，一个语句 S 中的所有变元都是局部的，即将每个子句视为其全称闭包，于是 S 就成了全称量词子句的合取式。因此，不同子句的变元之间没有任何关联。为了从语法角度描述这一点，我们在同时使用两个子句时通常会重命名变元，使得它们不含公共变元。(这个过程叫做*标准化变元分离*。)

和在命题逻辑中一样，至多含有一个正文字的子句叫做 Horn *子句*。其余术语，比如*程序子句*、*规则*、*事实*和*目标*(见第一章定义 10.4，或第二章定义 5.1)也都和命题逻辑中的定义相同。因此，例如，(PROLOG)*程序*，就是仅含程序子句的公式，程序子句是指恰好含有一个正文字的子句。下面继续使用 PROLOG 符号。

例 13.1 考虑下面子句的列表：

```
parent(X, Y):- mother(X, Y).                (1)
parent(X, Y):- father(X, Y).                (2)
daughter(X, Y):- mother(Y, X), female(X).   (3)
son(X, Y):- mother(Y, X), male(X).          (4)
child(X, Y):- son(X, Y).                    (5)
child(X, Y):- daughter(X, Y).               (6)
daughter(X, Y):- father(Y, X), female(X).   (7)
son(X, Y):- father(Y, X), male(X).          (8)
male(jim).                                  (9)
male(tim).                                  (10)
female(jane).                               (11)
female(pam).                                (12)
father(jim, tim).                           (13)
father(jim, pam).                           (14)
mother(jane, tim).                          (15)
mother(jane, pam).                          (16)
```

这些子句的 PROLOG 版本如下：

$$\{\{\mathrm{parent}(x, y), \neg\mathrm{mother}(x, y)\},$$
$$\{\mathrm{parent}(x, y), \neg\mathrm{father}(x, y)\},$$
$$\vdots$$
$$\{\mathrm{mother}(\mathrm{jane}, \mathrm{pam})\}\}.$$

也可以写成下面的子句形式：

$$\forall x \forall y[\mathrm{mother}(x,y) \to \mathrm{parent}(x,y)] \wedge$$
$$\forall x \forall y[\mathrm{father}(x,y) \to \mathrm{parent}(x,y)] \wedge$$
$$\cdots \wedge$$
$$\cdots \wedge$$
$$\cdots \wedge$$
$$\mathrm{mother}(\mathrm{jane},\mathrm{pam})。$$

定义 13.2 假设可以重命名 C_1 与 C_2 中的变元，使得它们不含相同变元且分别形如 $C_1'\sqcup\{P\vec{t}_1,\cdots, P\vec{t}_n\}$，$C_2'\sqcup\{\neg P\vec{s}_1, \cdots, \neg P\vec{s}_m\}$。如果 σ 是 $\{P\vec{t}_1, \cdots, P\vec{t}_n, P\vec{s}_1, \cdots, P\vec{s}_m\}$ 的一个 mgu，那么 $C_1'\sigma\cup C_2'\sigma$ 是 C_1 与 C_2 的**消解式**。($C_1'\sigma\cup C_2'\sigma$ 也叫做**父子句** C_1 和 C_2 的**儿子句**。)

C 的从 S 出发的消解证明以及 S 的消解反驳，不论是线性还是树形，都和命题逻辑的定义(第一章定义 8.4 和定义 8.6)类似。不同之处在于我们采用的消解规则是上述给定的版本，而且对任意改名替换 σ 以及任意 $C\in S$，允许取自 S 的前件(或等价地证明树的叶子)为 $C\sigma$。类似地，对于由 S 中元素的所有重命名构成的集合，定义 $\mathcal{R}(S)$ 为该集合下的消解闭包。

在上述消解式的定义中，有两点应当注意。第一，变元的重命名是必要的。例如，语句 $\{\{P(x)\}, \{\neg P(f(x))\}\}$(不可满足并且)消解可反驳，但是，如果不重命名变元，子句就无法合一。第二，在上述定义中不能像命题逻辑中那样假设 n 或 m 为 1。必须一次消除若干文字。(这一过程通常叫做因子分解(factoring)。)例如，$S=\{\{P(x), P(y)\}, \{\neg P(x), \neg P(y)\}\}$(不可满足并且)消解可反驳，但是任何从 S 出发且一次仅消除一个文字的消解证明都无法演绎出。

例 13.3 (i)通过消解

$$C_1 = \{Q(x), \neg R(y), P(x,y), P(f(z), f(z))\}$$

与

$$C_2 = \{\neg N(u), \neg R(w), \neg P(f(a), f(a)), \neg P(f(w), f(w))\}$$

来得到

$$C_3 = \{Q(f(a)), \neg R(f(a)), \neg N(u), \neg R(a)\}$$

为达此目的，首先通过 mgu $\{x/f(a), y/f(a), z/a, w/a\}$ 来合一 $\{P(x, y), P(f(z), f(z)), P(f(a), f(a)), P(f(w), f(w))\}$，然后对 C_1 和 C_2 作适当的替换和并进行消解。

(ii)从与例 13.1(3)和(16)相对应的子句出发，可以通过替换 $\{X/\mathrm{pam}, Y/\mathrm{jane}\}$ 得到消解式 $\{\mathrm{daughter}(\mathrm{pam}, \mathrm{jane}), \neg\mathrm{female}(\mathrm{pam})\}$。

例 13.4 (i)从下面的(a)和(b)出发，我们希望得到(c)：

(a) $\forall x\forall y\forall z[P(x, y)\wedge P(y, z)\to P(x, z)]$(传递性)

(b) $\forall x\forall y[P(x, y)\to P(y, x)]$(对称性)

(c) $\forall x \forall y \forall z[P(x,\ y) \wedge P(z,\ y) \rightarrow P(x,\ z)]$。

在子句形式下，我们希望从 $S=\{C_1,\ C_2\}$ 出发推出 C_3，这里

$$C_1=\{\neg P(x,y),\neg P(y,z),P(x,z)\},$$
$$C_2=\{\neg P(u,v),P(v,u)\},$$
$$C_3=\{\neg P(x,y),\neg P(z,y),P(x,z)\}。$$

（注意，我们已经对 S 中的子句作了标准化分离。）

下面给出一个消解树证明，树枝上标出了消解用到的替换。为明确起见，我们还用下划线标出了要消解的文字。

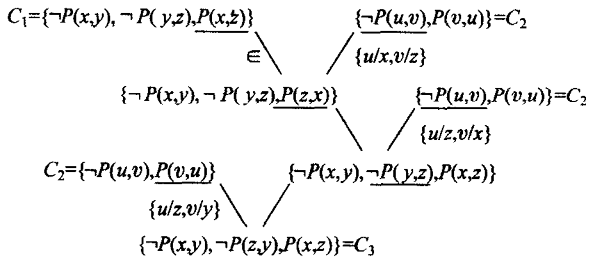


图 35

(ii)用下面的消解证明来说明：son(tim, jim)是由例 13.1 中的子句消解得到的。

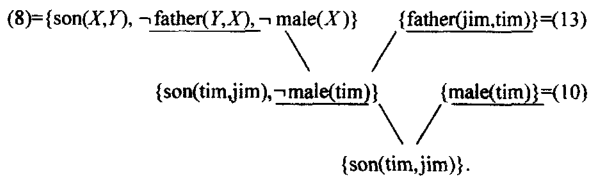


图 36

在上面提到的这些例子中，我们可以利用 PROLOG 来询问消解得到的结果。如果数据库中存有例 13.1 中的子句，可以在“?-”提示符处输入要问的子句“?-son(tim, jim).”，并且得到回答“yes”。PROLOG 将该问题解释成一个请求：从数据库出发证明 son(tim, jim)。更准确地说，如果在“?”提示符处输入正文字 C_1，C_2，…，C_n，PROLOG 就尝试从数据库 S 和目标子句 $G=\{\neg C_1,\ \cdots,\ \neg C_n\}$ 出发演绎出□。这里目标子句 G 也可以写成 $:-C_1,\ C_2,\ \cdots,\ C_n$。（注意，目标子句是指不含正文字的子句，即形如 $:-A_1,\ \cdots,\ A_n$，这里 A_i 都是正文字。由上述这些例子容易体会出目标子句等术语叫法的由来。）

如果 C_i 是基本项，那么我们期望得到□的一个从 $S\cup\{\neg C_1,\ \cdots,\ \neg C_n\}$ 出发的成功演绎，并据此推出所有 C_i 都是 S 的后承。（推理过程可以借助谓词逻辑中的消解可靠性，即下面的定理 13.6。在 C_i 都是基本项的情形下，我们也可认为该推理过程是由命题逻辑中的消解可靠性得到。）不过如果在“?”提示符处输入一个含有自由变元的原子公式，比如“male(X), female(Y)”，PROLOG 仍然将其视作一个从 S 和目标子句 $\{\neg \mathrm{male}(X),\ \neg \mathrm{female}(Y)\}$ 出发证明□的请求。这里的成功仅指从 S 出发得出结论 $\neg\forall X\forall Y[\neg \mathrm{male}(X)\vee\neg \mathrm{female}(Y)]$，因为子句 $\{\neg \mathrm{male}(X),\ \neg \mathrm{female}(Y)\}$ 是指 $\forall X\forall Y[\neg \mathrm{male}(X)\vee\neg \mathrm{female}(Y)]$。也就是说，得出结论 $\exists X\exists Y[\mathrm{male}(X)\wedge \mathrm{female}(Y)]$。而实际上 PROLOG 所作的是返回一个替换，比如 $X=\mathrm{jim}$，$Y=$

jane，这也验证了结论 $\exists X\exists Y[\text{male}(X)\wedge\text{female}(Y)]$ 的真假取决于 S 中的信息，即 $\{\neg\text{male(jim)},\neg\text{female(jane)}\}$ 与 S 不相容。当然，在实际应用中我们真正需要的就是这个正确的回答替换，而非一般性的断言 $\exists X\exists Y[\text{male}(X)\wedge\text{female}(Y)]$。

定义 13.5 如果 P 是一个程序，$G=\{\neg A_1,\cdots,\neg A_n\}$ 是一个目标子句，我们说(对于 G 中变元的)替换 θ 是一个**正确的回答替换**(correct answer substitution)，如果$(A_1\wedge A_2\wedge\cdots\wedge A_n)\theta$ 是 P 的一个逻辑后承(也即是 P 的全称闭包的逻辑后承)。

注意，根据第十节习题5中厄布朗定理的一个应用，如果 $P\cup\{G\}$ 是不可满足的，那么存在一个基本替换是正确的回答替换。我们总能通过消解找到这样一个替换，这就是完全性定理要表述的思想。我们在证完可靠性定理之后再回来考虑消解的完全性。在本节及下一节讨论一般的消解方法，关于 PROLOG 中要用的特殊搜索程序的讨论放在第三章。

定理 13.6(消解的可靠性) 如果$\Box\in\mathcal{R}(S)$，那么 S 是不可满足的。

证明 为了导出矛盾，假设$\mathcal{A}\vDash S$。消解的记法依照定义 13.2。我们要证明的是：如果$\mathcal{A}\vDash C_1$，C_2 且 C 是 C_1，C_2 的消解式，那么$\mathcal{A}\vDash C$，即$\mathcal{A}\vDash C\tau$ 对每一个基本替换 τ 都成立。(若如此，则可以归纳证明$\mathcal{A}\vDash C$ 对每一个 $C\in\mathcal{R}(S)$成立。又由于$\mathcal{R}(S)$包含$\Box$，所以导出矛盾。)这里唯一需要注意的是：若$\mathcal{A}\vDash C_i$，则$\mathcal{A}\vDash C_i\sigma_i$ 对任意 σ_i 成立，因为 C_i 是开公式。对于 $C=C_1'\sigma\cup C_2'\sigma$ 中变元的每一个基本替换 τ，我们可以按照命题逻辑的情形来讨论。(见第一章引理 8.12 及定理 8.11。)由于对每个基本替换 τ，$C_1'\sigma\tau$ 或 $C_2'\sigma\tau$ 在$\mathcal{A}$中为真(哪个为真取决于要消解的文字在$\mathcal{A}$中的真假以及该文字在哪个 $C_i'\tau$ 中作为正文字出现)，所以它们的并集 $C\tau$ 在$\mathcal{A}$中为真。 □

现在将谓词逻辑中消解方法的完全性证明归约到命题逻辑的情形。首先给出两个引理。第一个引理(引理 13.7)是将命题逻辑和谓词逻辑中的单步消解联系起来，而第二个引理(引理 13.8)则将相应的结果从单步消解推广到了整个消解证明。引理 13.8(经常被称作提升引理，因为它将命题逻辑中的证明“提升”到了谓词逻辑)对于分析消解的各种限制版本非常有用，这在第十四节以及第三章第一节都有体现。由于$\Box$的消解证明用处尤其广泛，所以我们将其作为推论 13.9 单独列出。

引理 13.7 如果 C_1'，C_2'分别为 C_1，C_2(通过替换 θ_1，θ_2 得到)的基本实例，C'是 C_1'与 C_2'的消解式，那么存在 C_1 与 C_2 的消解式 C，使得 C'是 C 的一个基本实例(如果 C_1，C_2 不含公共变元，那么该实例可以通过 $\theta_1\theta_2$ 得到)。

证明 由于消解规则允许重命名 C_1，C_2 中的变元，所以不妨设 C_1，C_2(因而 θ_1，θ_2 也)不含公共变元。由于 $C_1'=C_1\theta_1$ 与 $C_2'=C_2\theta_2$ 可消解，例如对基本文字 $P(t_1,\cdots,t_n)$可消解，那么存在文字集合

$$A_1=\{P(\vec{s}_{1,1}),\cdots,P(\vec{s}_{1,n_1})\}\subseteq C_1$$

和

$$A_2=\{\neg P(\vec{s}_{2,1}),\cdots,\neg P(\vec{s}_{2,n_2})\}\subseteq C_2$$

通过替换 θ_1，θ_2 分别合一为$\{P(t_1,\cdots,t_n)\}$和$\{\neg P(t_1,\cdots,t_n)\}$。由于分别由 θ_1 和 θ_2 中变元构成的集合不相交，所以 $\theta_1\theta_2$ 同时合一文字集合 A_1 和 A_2。因此，根据谓词演算中的消解定义(定义 13.2)，$C=((C_1-A_1)\cup(C_2-A_2))\sigma$ 是 C_1 与 C_2 的消解式，这里 σ 是

$$\{\neg P(\vec{s}_{1,1}),\cdots,\neg P(\vec{s}_{1,n_1})\}\cup\{\neg P(\vec{s}_{2,1}),\cdots,\neg P(\vec{s}_{2,n_2})\}$$

的 mgu，由合一算法给出。接下来只需要验证 C'是 C 的一个实例。我们断言 $C'=C\theta_1\theta_2$。注

意，由于 $\theta_1\theta_2$ 合一了 $\neg A_1 \cup A_2$，合一算法(定理 12.5)给出的 mgu 所具有的特殊性质保证了 $\sigma\theta_1\theta_2=\theta_1\theta_2$。因此

$$
\begin{aligned}
C\theta_1\theta_2 &= ((C_1-A_1)\cup(C_2-A_2))\sigma\theta_1\theta_2\\
&= ((C_1-A_1)\cup(C_2-A_2))\theta_1\theta_2\\
&= (C_1\theta_1-A_1\theta_1)\cup(C_2\theta_2-A_2\theta_2)\text{(根据变元的分离性)}\\
&= (C'_1-\{P(t_1,\cdots,t_n)\})\cup(C'_2-\{\neg P(t_1,\cdots,t_n)\})\\
&= C'\text{(根据定义)}
\end{aligned}
$$

□

引理 13.8(提升引理) 令 S 为语言$\mathcal{L}$中的一个公式，S'为 S 中的子句在$\mathcal{L}$的厄布朗域中的所有基本实例构成的集合。如果 T'是 C'的一个从 S'出发的消解树证明，那么存在$\mathcal{L}$的一个子句 C，C 的一个从 S 出发的消解树证明 T 以及一个替换 θ，使得 $T\theta=T'$(即 T 和 T'标识同一棵树且 $C_i\theta=C'_i$，C_i，C'_i是同一棵树上的每个节点分别在 T 和 T'下的标签。因此，$C'=C\theta$)。此外，如果 T'的叶节点标签为 R_i 并且每个 R_i 都是 S 中的某个 S_i 的一个实例，那么我们可以将 T 中相应的叶节点标识为适当的 S_i 的改名。

证明 对从 S'出发的消解树证明的深度做归纳。对于基本情形，R_i 是 S'中的元素，由于每个 R_i 都是 S 中某个元素的一个替换实例，引理得证。现在考虑 C'的一个从 S'出发深度为 $n+1$ 的证明。该证明由两部分组成：首先是基本子句 C'_1，C'_2的从 S'出发(深度$\leqslant n$)的消解树证明 T'_1，T'_2，然后消解 C'_1，C'_2得到 C'。假设 $P(t_1,\cdots,t_n)\in C'_1$，$\neg P(t_1,\cdots,t_n)\in C'_2$，对该文字进行消解得到

$$C' = C'_1\cup C'_2-\{P(t_1,\cdots,t_n),\neg P(t_1,\cdots,t_n)\}$$

由归纳假设，我们有谓词子句 C_1，C_2 和 C_1，C_2 的证明树 T_1，T_2 以及替换 θ_1，θ_2，使得 $T_i\theta_i=T'_i$。(T_i 的叶节点同样已由归纳假设标识出来。)由于可以对 T_1，T_2 中的变元进行改名替换，所以不妨设 θ_1，θ_2 不含公共变元。(因为消解规则允许对父子句作任意改名替换，所以 T_i 能够保持消解证明。由于引理只要求叶节点标签被 S 中子句的某个改名所标识，所以这个改名并不影响叶节点被适当标识。)应用引理 13.7 得到 C_1 与 C_2 的消解式 C，并且 $C'=C\theta_1\theta_2$。现在通过结合 T_1，T_2，对 C_1 与 C_2 进行消解得到 C 的一个从 S 出发的消解树证明 T。由于 θ_1 和 θ_2 不相交，所以 $T\theta_1\theta_2$ 限制在 T_1，T_2 上分别为 $T_1\theta_1$，$T_2\theta_2$。显然，对于 T 的剩余节点 C，有 $C\theta_1\theta_2=C'$。因此 T 是 C 的从 S'出发的谓词逻辑消解证明，正如所求，而 $\theta_1\theta_2$ 即是这个引理要找的替换。 □

推论 13.9 如果 T'是□的一个消解树证明，并且树上每个叶子 L_i 被子句 S_i 的一个基本实例 R_i 所标识，那么存在□的一个消解树证明 T，它与 T'树形相同并且每个叶节点 L_i 被 S_i(的一个改名)所标识。

证明 这只是引理 13.8 的一种特殊情形，即 $C'=$□。唯一需要注意的是，只有□子句本身才具有替换实例□。 □

定理 13.10(消解的完全性) 如果 S 是不可满足的，则□$\in\mathcal{R}(S)$。

证明 令 S'为 S 中的子句在$\mathcal{L}$的厄布朗域中的所有基本实例构成的集合。根据厄布朗定理的一个推论(定理 10.6)，S 与 S'是可满足等价的。因此，如果假设 S 是不可满足的，那么 S'亦是不可满足的。由命题逻辑中的消解完全性(第一章定理 8.15 或定理 8.22)知，□$\in\mathcal{R}_p(S')$，这里用$\mathcal{R}_p$来表示命题逻辑中的消解程序。(在此情形下通常将原子公式视作命题字

母。)现在利用推论 13.9，立即得到谓词逻辑中的消解完全性(即如果 S 是不可满足的，则 $\square \in \mathcal{R}(S)$)。 $\square$

习题

1. 寻找消解式：

a) $\{P(x, y), P(y, z)\}$，$\{\neg P(u, f(u))\}$

b) $\{P(x, x), \neg R(x, f(x))\}$，$\{R(x, y), Q(y, z)\}$

c) $\{P(x, y), \neg P(x, x), Q(x, f(x), z)\}$，$\{\neg Q(f(x), x, z), P(x, z)\}$。

2. 把下列语句转换成谓词逻辑，然后用子句形式表示出来并利用消解方法证明结论：

a)假设所有理发师都给那些不自己理发的人理发。此外，没有理发师给那些自己理发的人理发。推出结论：不存在理发师。

b)假设 John 喜欢那些自我厌恶的人。推出结论：这与“John 不喜欢任何自我欣赏的人”不是同一种情形。

3. 假设我相信下列所有四个表述：

(i)世上有龙。

(ii)龙要么在洞穴睡觉，要么在森林猎食。

(iii)如果龙饿了，它就不能睡觉。

(iv)如果龙累了，它就不能猎食。

将以上四个表述转化成谓词逻辑。利用消解回答下列问题：

(a)如果龙饿了，它会做什么?

(b)如果龙累了，它会做什么?

(假设如果 X 不能做 Y，那么 X 就不会去做 Y。)

4. (a)用子句形式表示下列三个表述：

(i)人人仰慕英雄。

(ii)失败者仰慕所有人。

(iii)非英雄即是失败者。

(b)利用消解寻找 X，Y，使得他们互相仰慕。

5. 给出下列子句集合的消解反驳。标出每一步所需的替换以及要消解的文字：

(i) $\{P(a, x, f(y)), P(a, z, f(h(b))), \neg Q(y, z)\}$

(ii) $\{\neg Q(h(b), w), H(w, a)\}$

(iii) $\{\neg P(a, w, f(h(b))), H(x, a)\}$

(iv) $\{P(a, u, f(h(u))), H(u, a), Q(h(b), b)\}$

(v) $\{\neg H(v, a)\}$。

6. 考虑下面七个语句。

(i)即将拥有房地产合伙人的那些股东会投票反对提案，而其他人不会。

(ii)john 和 jim(同样地 mary 和 jane)将会建立房地产合伙人关系，如果某家银行愿意为他们提供贷款，并且有律师能帮助他们拿到所需的区域许可证。

(iii)如果没有律师能够为他们拿到所需的区域许可证，那么就没有银行愿意为其提供贷款来建立房地产合伙人关系。如果有了区域许可证，那么再有一个良好的资产评估银行就同意贷款。

(iv)john 和 jane 都是股东。

(v)joyce 是一名律师，只要你有足够多的资金，他就能够为你拿到区域许可证。

(vi)john 拥有巨额资金，他和 jim 的土地已经获得良好的资产评估。

(vii)有一家银行。

将以上语句转化成谓词逻辑，用子句形式表示出来，并利用消解方法推出：某人将会投票反对提案。他是谁?

第十四节　加细消解：线性消解

在生成消解证明的过程中，系统尝试常常会显得冗赘低效。和命题情形一样，我们可以对消解做各种限制，使其更加高效。(与第一章第九节情况类似的)各种加细消解在习题中都有所体现。第一章第十节对于命题 Horn 子句的消解作了加细，即线性消解。现在我们希望在完全谓词逻辑中分析线性消解。基本思路是通过消解的一个线性序列而非分叉树来进行演绎。我们挑选这样一列消解，使得(在第一步之后)每一步消解有一个父节点是前一步消解的儿节点。

定义 14.1　令 C 是子句，S 是公式。

(i) C 的一个**从 S 出发的线性演绎**是指子句序对的一个序列 $\langle C_0, B_0\rangle, \cdots, \langle C_n, B_n\rangle$，满足 C_0 和每个 B_i 要么是 S 中元素的改名替换，要么是某个 C_j，$j<i$；每个 $C_{i+1}(i\leqslant n)$ 是 C_i 与 B_i 的消解式；且 $C_{n+1}=C$。

(ii) C 是**从 S 出发线性可演绎的**，记作 $S\vdash_{\mathcal{L}} C$，如果存在 C 的一个从 S 出发的线性演绎。存在 S 的一个**线性消解反驳**，如果□是从 S 出发线性可演绎的。$\mathcal{L}(S)$ 是由那些从 S 出发线性可演绎的所有子句组构成的集合。

我们画出了一个线性消解，如图 37 所示。

下面的这个例子曾在第一章例 10.2 出现，它的线性消解如图 38 所示。

$$S=\{A_1,A_2,A_3,A_4\},A_1=\{p(x),q(x)\},A_2=\{p(x),\neg q(x)\},$$
$$A_3=\{\neg p(x),q(x)\},A_4=\{\neg p(x),\neg q(x)\}。$$

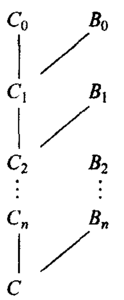


图 37

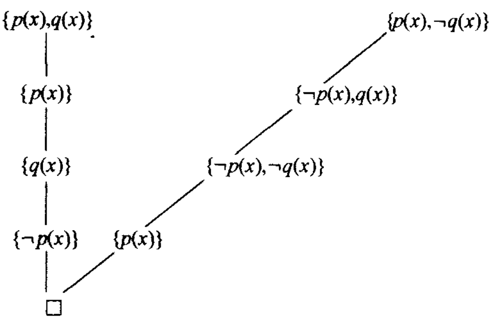


图 38

定义 14.2　在线性消解中，S 中的元素通常称作**插入子句**，而 C_i 和 B_i 分别被称作**中间子句**和**边子句**。

下面扩展“父-儿”术语系统：在 C 的从 S 出发的消解证明中，将子句 C 的祖先定义为证明树中在 C 之上的所有子句。这样一来我们就能重述线性演绎的定义如下：每个 C_i 都是与其祖先或某个插入子句进行消解，得到 C_{i+1}。

我们希望证明线性消解是完全的。(由于线性消解是一般消解的限制版本，因此其可靠性不证自明。)为了最后的归纳推理，也为了将线性消解应用到 PROLOG 当中，我们实际上证明了一个更强的结果：它对线性证明的起始点(即是上面提到的 C_0)作了一些限制。

定义 14.3　$U\subseteq S$ 是 S 的一个**支集**(set of support)，如果 $S-U$ 是可满足的。在此情形下我们说 C 的从 S 出发的线性消解证明 $\langle C_i, B_i\rangle(i\leqslant n)$ 有**支集** U，如果 $C_0\in U$。

该定义的直观意义如下：对于公式 $S\in$ UNSAT，导致 S 是不可满足的“因素”全部包含在 U 中（U 对 $S\in$ UNSAT 起到“支撑”作用）。

现在可以表述完全性定理的一个加强版本。

定理 14.4 如果 $S\in$ UNSAT，U 是 S 的一个支集，那么存在 S 的一个带有支集 U 的线性反驳。

首先要做的是将定理 14.4 的证明归约到下述情形：S 的所有非空子集都是其支集。

定义 14.5 S 是**极小不可满足的**（minimally unsatisfiable），如果它是不可满足的而其所有真子集是可满足的，即对每一个 $C\in S$，$\{C\}$ 是 S 的一个支集。

引理 14.6 如果 $S\in$ UNSAT，那么存在一个极小不可满足的 $S'\subseteq S$。此外，如果 U 是 S 的一个支集，那么 $U\cap S'$ 是 S' 的一个支集。

证明 由紧致性知，S 的某个有穷子集是不可满足的。如果 S' 是 S 的包含最少子句的一个不可满足子集，那么 S' 肯定是 S 的一个极小不可满足子集。令 U 为 S 的任一支集。如果 $U\cap S'=\varnothing$，那么 S' 就包含在可满足集合 $S-U$ 当中，导出矛盾。因此，$S'-(S'\cap U)$ 是 S' 的一个真子集，再根据 S' 的极小性可以得到，$S'-(S'\cap U)$ 是可满足的。 □

证明（定理 14.4） 我们这里的计划仍然是将证明归约到命题逻辑的情形。（至于命题逻辑的情形，在后面会给出证明。）和处理一般消解的手法相同，利用厄布朗定理。如果 S 不可满足且有支集 U，则由 S 中元素的所有基本实例构成的集合 S' 也不可满足。S' 有支集 U'，U' 是由 U 中元素的所有基本实例构成的集合。我们希望证明，□的从 S' 出发带有支集 U' 的任何线性消解证明 T'，都可以提升到□的从 S 出发带有支集 U 的线性消解证明。这可由提升引理的推论 13.9 直接得到。提升引理能够保持消解树的形状，所以它将线性证明提升到线性证明。对于树叶上的子句 S_i，提升引理将其实例 R_i 提升到 S_i（的改名）。因此，如果 T' 中的子句 C_0' 在 U' 中，则 C_0' 是 U 中某个子句 C 的一个实例，从而 C_0' 被提升到子句 C（的一个改名）。 □

现在来证明命题演算中线性消解完全性定理的加强版本。这里的证明要比语义或序消解复杂得多，但它与支集以及第一章第九节习题 3 提到的锁定消解相关联。

证明（定理 14.4 的命题版本） 根据引理 14.6，只需考虑那些极小不可满足的 S。（根据定义，对于 $S'\subseteq S$，S' 的任何带有支集 $U\cap S'$ 的线性消解反驳都是 S 的带有支集 U 的线性消解反驳。）对 $E(S)$ 作归纳，$E(S)$ 代表 S *溢出文字数目*（excess literal number），即 S 中文字出现的数目减去 S 中子句的数目。（注意，在定义溢出文字数目时需要 S 有穷。）事实上我们要归纳证明：对任何 $C\in S$，存在 S 的一个由 C 开始的线性反驳，即在证明树中 $C=C_0$。对于基本情形，如果 $E(S)=-1$，$\square\in S$，无须证明。现在假设 $E(S)\geq 0$。

情形 1：C 是单元子句，即其恰好包含一个文字 ℓ。我们断言必定存在一个子句 $C'\in S$，使得 $\bar{\ell}\in C'$。若不然，则任何满足 $S-C$ 的指派（根据 S 的极小性易知，$S-C$ 是可满足的）加上 ℓ 之后都满足 S。注意 $\ell\notin C'$。因为如果 $\ell\in C'$，那么 C' 是重言式，S 就不是极小不可满足式，与假设矛盾。因此，由 S^{ℓ} 的定义（第一章定义 8.16）知，$C'-\{\bar{\ell}\}$ 在 S^{ℓ} 中。如果 $C'=\{\bar{\ell}\}$，则证明完成，因为我们可以对 C 与 C' 消解得到□。下面假设 $C'=\{\bar{\ell},\cdots\}$ 包含不止一个文字。由于 $S\in$ UNSAT，根据第一章引理 8.19，$S^{\ell}\in$ UNSAT。从 S 出发构造 S^{ℓ} 的过程中所消去的每个子句都至少包含一个文字（ℓ）（又是由定义得知）。因此，这些子句的消去并不能增加溢

出文字数目。另一方面，至少 C' 在此过程中损失了一个文字($\bar{\ell}$)。因此，$E(S^{\ell})<E(S)$。

下面断言：S^{ℓ} 也是极小不可满足的。假设 $D\in S^{\ell}$ 而 $S^{\ell}-\{D\}$ 是不可满足的。现在根据 S^{ℓ} 的定义，$D\in S$ 或者 $D\cup\{\bar{\ell}\}\in S$，在这两种情形下都有 $\ell\notin D$。令 D' 表示 S 中的任一子句。根据 S 的极小不可满足性知，$S-\{D'\}$ 是可满足的，令 $\mathcal{A}$ 为满足它的一个指派。由于 $C=\{\ell\}\in S-\{D'\}$，所以 $\mathcal{A}\vDash\ell$。现在考虑任意子句 $F\in S^{\ell}-\{D\}$ 以及相对应的 $F'\in S-\{D'\}$。由于 $\mathcal{A}\vDash\ell$，$\mathcal{A}\vDash F'$，所以在 F' 定义的各种情形下都有 $\mathcal{A}\vDash F$。(和从 D 出发定义 D' 一样，F' 是由 F 定义得来。)因此，$\mathcal{A}\vDash S^{\ell}-\{D\}$，与我们的假设矛盾。

我们的归纳假设给出了□的一个从 S^{ℓ} 出发由 $C'-\{\bar{\ell}\}$ 开始的线性消解演绎：$\langle C_0, B_1\rangle$，…，$\langle C_n, B_n\rangle$，其中 $C_0=C'-\{\bar{\ell}\}$。每个 B_i 要么是 S^{ℓ} 中的元素，要么是某个 $C_j(j<i)$。C_n 与 B_n 消解得到□。分段构造一个新的证明 $\langle D_j, A_j\rangle$，使得第 i 段以 $D_k=C_i$ 结尾。首先令 $D_0=\{\ell\}=C$ 且 $A_0=C'$。很明显它们可以消解得到 $D_1=C_0$。现在采用归纳法。假设已经有 A_j $(j<k)$ 并且 $D_k=C_i$。如果对某个 $j<i$，有 $B_i=C_j$，那么令 $A_k=C_j$(由归纳假设知，它是之前的一个 D_m)并且消解得到 $D_{k+1}=C_{k+1}$。否则，$B_i\in S^{\ell}$，我们要考虑两种情形。如果 $B_i\in S$，那么令 $A_k=B_i$ 并且消解得到 $D_{k+1}=C_{i+1}$。如果 $B_i\notin S$，那么 $B_i\cup\{\bar{\ell}\}\in S$，令 $A_k=B_i\cup\{\bar{\ell}\}$ 并且消解得到 $D_{k+1}=C_{i+1}\cup\{\bar{\ell}\}$。在此情形下，令 $A_{k+1}=\{\ell\}$ 并且消解得到 $D_{k+2}=C_{i+1}$，然后继续归纳过程。由于 $\{\ell\}=D_0$，所以就得到了 S 的一个要求的线性消解。

情形 2：$C=\{\ell, \cdots\}$ 多于一个文字的情形。现在考虑 $S^{\bar{\ell}}$。由情形 1 知，它是极小不可满足的并且其溢出文字数目少于 S。因此，根据归纳，有□的一个从 $S^{\bar{\ell}}$ 出发由 $C-\{\ell\}$ 开始的线性消解演绎。如果对每一个中间子句以及那些在 $S^{\bar{\ell}}$ 中但不在 S 中的边子句添加 ℓ，那么就得到了 $\{\ell\}$ 的一个从 S 出发由 C 开始的线性证明。现在考虑 $S'=S-\{C\}\cup\{\{\ell\}\}$，它也是不可满足的。(任何满足它的指派都满足 S。)由于 C 中不止一个文字，所以 $E(S')<E(S)$。现在由于 $\square\notin S'$，对任一 $S''\subseteq S'$，$E(S'')\leq E(S')$。如果取 $S''\subseteq S'$ 为极小不可满足子句，那么由归纳知，存在□的一个从 $S''\subseteq S\cup\{\{\ell\}\}$ 出发由 $\{\ell\}$ 开始的线性消解证明。(注意，根据 S 的极小不可满足性，$S'-\{\ell\}=S-\{C\}$ 是可满足的。因此，S' 的任何不可满足的子集 S'' 都必然包含 $\{\ell\}$)。将此证明连接到 $\{\ell\}$ 的某个从 S 出发的证明的末端，我们就得到所要求的 S 的由 C 开始的线性消解。 □

对于一般消解的加细还有很多，其中一些(比如序线性消解)将在习题中给出。后面我们不再宽泛地考虑一般性的问题，而是将重点放在消解在 Horn 子句中的特殊应用，即 PROLOG 的演绎机制上。

习题

1. 根据第一章第九节给出的命题逻辑中序消解的模型，完整定义谓词演算中的序消解和序消解反驳。注意，首先要对语言中的谓词符号排序。
2. 表述并证明：谓词逻辑中序消解的可靠性定理。
3. 表述并证明：谓词逻辑中序消解的完全性定理。
4. 定义并证明：谓词逻辑中 F－消解(第一章第九节习题 4)的可靠性和完全性定理。
5. 定义并证明：谓词逻辑中锁定消解(第一章第九节习题 1)的可靠性和完全性定理。

进一步阅读建议

要搞清谓词逻辑的起源，参阅 van Heijenoort[1967，2.1]中关于 Frege[1879]的部分。

要想更多了解表、公理与推理规则、消解、自然演绎和矢列式(sequent)的谓词逻辑版本，参阅第一章结尾处所列参考书目中的相关主题。

有关(超出本书范畴的)模型论的奠基性工作，可以参阅 Chang，Keisler[1990，3.4]或者参考文献列表 3.4 列出的任何一本教材。Hodges[1993，3.4]涵盖了当前该学科的很多内容。

要搞清厄布朗域和合一的来源，参阅 van Heijenoort[1967，2.1]中关于 Herbrand[1930]的前几页，或者参阅 Herbrand[1971，2.3]。

Chang，Lee[1973，5.7]以厄布朗定理为基础，阐述了数理逻辑的基本内容，这本书还介绍了多种类型的消解。关于厄布朗定理，也可以参阅 Loveland[1978，5.7]和 Bibel[1987，5.7]。

Lloyd[1987，5.4]是关于 PROLOG 理论的标准教材，它也给出了一个广泛的参考文献列表。在第三章的结尾会给出更多有关 PROLOG 的阅读建议。

第三章　PROLOG

第一节　SLD-消解

本章考虑用于谓词逻辑式编程的 PROLOG 语言。许多基本术语不过是第一章第十节所介绍的基本术语的谓词逻辑版本。尽管如此，为了在谓词演算中证明消解定理，我们仍以适当的形式重述这些基本定义。虽然 PROLOG 采用加细的线性消解，但是我们对它所做的论述都独立于线性消解(相当困难)的完全性定理(第二章定理 14.4)。不过，我们假设读者熟悉第二章中线性消解的相关定义 14.1 ~ 14.3。因此我们的证明是以第一章第十节所探讨的命题版本的 PROLOG 为基础，同时应用厄布朗定理(第二章定理 10.4)和该定理所给出的谓词逻辑向命题逻辑的归约。如果能用第二章第十四节的知识说明某些观点或者简化某些证明，我们就用 * 标出这个替代的结果或证明。

回顾第二章定义 5.1，一个 PROLOG 程序 P 就是一个程序子句的集合，所谓程序子句就是有且只有一个正文字的子句。我们在提示符“?-”下输入一串正文字 A_1, …, A_n 进行提问，PROLOG 编译器把输入转换成一个目标子句 $G=\{\neg A_1, \cdots, \neg A_n\}$，然后通过检验 $P\cup\{G\}$ 是否是不可满足的来回答问题。下面我们来描述 PROLOG 如何检验 $P\cup\{G\}$ 是否是不可满足的。从第一章第十节中为命题逻辑引入的线性消解方法开始，第二章第十四节已经证明该方法对谓词逻辑是完全的。下面仅考虑线性消解的插入版本。一般来说这种版本的消解不是完全的(由第二章定义 14.1 之后的例子即可看出)，但它在 PROLOG 环境中被证明是完全的。在本节的剩余部分，设 P 是一个 PROLOG 程序，G 是一个目标子句。

定义 1.1　*线性消解证明* $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle, G_{n+1}$ *是* $P\cup\{G\}$ *的一个***线性插入(消解)反驳**或 **LI(消解)反驳**，*如果* $G_0=G$, $G_{n+1}=\Box$, G_i *是目标子句且* C_i *是* P *中子句(的改名)。*

***定理 1.2**　*如果* $P\cup\{G\}\in\mathrm{UNSAT}$，*那么* $P\cup\{G\}$ *有一个线性插入消解反驳。*

***证明**　因为每一个 PROLOG 程序都是可满足的(第二章第五节习题 6)，所以 $\{G\}$ 是 $P\cup\{G\}$ 的支集。根据线性消解完全性定理(第二章定理 14.4)，存在一个线性反驳 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$，其中 $G_0=G$。我们断言：每个 G_i 是一个目标子句，并且如果不考虑命名方面的差异，则每个 $C_i\in P$。在归纳证明的过程中唯一需要注意的是，两个目标子句无法进行消解，而目标子句与程序子句消解的结果要么是目标子句，要么是□，因此下一个 C_i 必然来自 P。　□

我们现在知道 PROLOG 消解证明的一般模式：线性插入消解。在继续考察其他通过 PROLOG 执行的加细消解之前，我们应当注意到(所有消解方法的执行通常都是如此)，比如说，当 $G=\{\neg A_0, \cdots, \neg A_n\}$ 与 $C=\{B, \neg B_0, \cdots, \neg B_m\}$ 在 $A_i=B$ 上通过 mgu θ 进行消解时，编译器不会检查结果中是否有要消除的重复(例如，$A_j\theta=A_\ell\theta$ 或 $A_j\theta=B_\ell\theta$ 或 $B_k\theta=B_\ell\theta$)。它只是用 $\neg B_0, \cdots, \neg B_m$ 取代 $\neg A_i$，然后把 θ 作用到其中各项，不做进一步的简化。为理解这些执行，我们应当(像机器那样)把子句看成*有序子句*(ordered clause)，即文字序列，而非文字集合。如此一来，上面所说的消解就是用 $\neg B_0, \cdots, \neg B_m$ 替换 $\neg A_i$ 并把 θ 作用到序

列的每个文字上，从而得到下一个(有序)目标子句。子句的顺序不会对消解结果造成严重的影响。下面的定义和引理将说明这一点。

定义 1.3

(i)如果 $G=\{\neg A_0, \cdots, \neg A_n\}$ 和 $C=\{B, \neg B_0, \cdots, \neg B_m\}$ 是有序子句且 θ 是 A_i 和 B 的一个 mgu，那么可以在文字 A_i 上对 G 和 C 做**有序消解**。这个消解的**(有序)消解式**((ordered) resolvent)是有序子句 $\{\neg A_0, \cdots, \neg A_{i-1}, \neg B_0, \cdots, \neg B_m, \neg A_{i+1}, \cdots, \neg A_n\}\theta$。

(ii)如果 $P\cup\{G\}$ 是给定的有序子句集合，那么 $P\cup\{G\}$ 的**线性确定反驳**(linear definite refutation)或者 **LD-反驳**是有序子句 G_i，C_i 的序列 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$，其中 $G=G_0$，$G_{n+1}=\Box$，每个 G_i 是一个有序目标子句，每个 C_i 是 P 的某个子句的改名，且该子句不含 $G_j(j\leqslant i)$ 或 $C_k(k<i)$ 中的变元，而每个 $G_{i+1}(0\leqslant i\leqslant n)$ 是 G_i 与 C_i 的一个有序消解式。如果 C_n 不为$\Box$，我们就称该序列是一个 **LD-消解证明**。

注意，这个方法没有采用合并文字的策略。我们从每个子句中选出一个文字进行消解，并在消解式中删除且仅删除这两个文字。

引理 1.4(LD-消解的完全性)　如果 $P\cup\{G\}$ 是有序子句构成的不可满足集合，那么存在 $P\cup\{G\}$ 的一个由 G 开始的 LD-反驳。

证明　考虑 $P\cup\{G\}$ 在适当的厄布朗域中的(有序)子句的所有基本实例 $P'\cup G'$。由厄布朗定理(第二章定理 10.6)知 $P'\cup G'$ 是不可满足的，再由紧致性定理(第二章定理 7.9)，存在 $P'\cup G'$ 的某个有穷子集是不可满足的。因为所有程序子句集合均是可满足的(第二章第五节习题 6)，所以任何这样的子集都必包含 G' 的元素，即 G 的实例。令 $P''\cup G''$ 为 $P'\cup G'$ 的一个极小不可满足子集。根据极小性，存在 $G''_0\in G''$，使得 $P''\cup G''-\{G''_0\}$ 是可满足的。根据第一章引理 10.11，存在 $P''\cup G''$ 的一个由 G''_0 开始的 LD-消解反驳。利用下面的引理 1.5，可以将该 LD-反驳提升为 $P\cup G$ 的 LD-反驳。 $\Box$

***证明**　令 P' 和 $\{G'\}$ 分别为对应于 P 和 $\{G\}$ 的无序子句集合，对 $P'\cup\{G'\}$ 的 LI-反驳的长度作归纳。值得注意的是，可以用一个 LD-消解序列来替换一个 LI-消解，这样也可以实现 LI-消解中所进行的文字合并。证明细节留作习题 1~2。 $\Box$

引理 1.5　提升引理对 LD-消解证明成立。更确切地说，第二章引理 13.7 对有序消解成立，第二章引理 13.8 对 LD-消解证明成立，第二章推论 13.9 对 LD-消解反驳成立。

证明　该证明本质上与第二章第十三节的证明相同。不过单个消解(第二章引理 13.7)的提升要简单一些，因为这里不涉及文字合并(所以那个证明中的参数 n_1 和 n_2 都等于 1)。在那个提升引理本身的证明中，我们注意到，对线性消解来说，对树的深度作归纳等同于对证明长度作归纳。树叶是起始子句和边子句。所有与第二章第十三节雷同的证明细节留作习题 3~5。 $\Box$

下面描述在 LD-消解证明中，应当如何从 G_i 中选取用于消解的文字。几乎所有 PROLOG 执行都使用如下选择规则，即始终消解 G_i 中第一个(最左边的那个)文字。消解式中来自 C_i 的文字仍保持其原来的顺序并且放在所有来自 G_i 的子句的左边。我们称之为 SLD-消解。(S 代表 selection。)更一般地，可以考虑任意选择规则 R，即任何从每个有序目标子句中选择一个文字的函数。

定义 1.6　一个**选择规则** R 就是从每个非空有序子句 C 中选择一个文字 $R(C)$ 的函数。$P\cup\{G\}$ 的一个**带 R 的 SLD-反驳**是 $P\cup\{G\}$ 的一个 LD-反驳 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$，其

中 $R(G_i)$ 是该证明第 i 步所消解的文字。(如果没有指明 R，就默认采用标准选择规则，即总是选择最左边文字进行消解。)

下一个目标是证明 SLD－反驳的一个完全性定理。首先沿着前面的论证思路，应用关于 SLD－证明的提升引理给出一个简单证明。在命题逻辑的层面上，该证明的核心在本质上就是第一章定理 10.13，即命题逻辑的 SLD－消解完全性定理的证明。将该结果直接提升到谓词逻辑的困难在于，提升引理不能直接应用于带任意选择规则的 SLD－证明。这是因为，选择规则从某个子句的谓词提升中选取的文字和它从该子句的基本实例中所选文字的提升并不一定相同。幸运的是，这个问题对于一大类选择规则而言并不存在，包括总是选取最左边文字的标准选择规则。因而，对于我们所感兴趣的那些可用于 PROLOG 执行的选择规则，可以直接提升命题版本的完全性定理。

定义 1.7 选择规则 R 是**恒定的**(invariant)，如果对每个(有序)子句 C 和每个替换 θ，$R(C\theta)=(R(C))\theta$。

注意，标准选择规则显然是恒定的。

定理 1.8(SLD－反驳的完全性) 如果 $P\cup\{G\}\in$ UNSAT，则对任意选择规则 R，存在$P\cup\{G\}$ 的一个带 R 的 SLD－消解反驳。

证明(对于恒定选择规则) 除了用第一章定理 10.13 替换第一章引理 10.11 之外，我们的论证与引理 1.4 的证明完全相同。然后应用带恒定选择规则的 SLD－消解的提升引理(习题 6)即可。 □

对于任意选择规则，现在可以给定理 1.8 提供一个直接但是相当复杂的证明。为了简化证明，先证明一个引理，该引理断言一个独立性结果：给定一个由 G 开始的 LD－反驳，可以找到一个带任意选择规则的 SLD－反驳。然后由引理 1.4 可以直接得到定理 1.8 的一般形式。这个独立性结果的证明是需要技巧的，我们把它推迟到引理 1.12。

我们现在知道，以“? $-A_1$，…，A_n.”输入一个问题时 PROLOG 编译器需要做什么。它从当前的程序 P 和目标子句 $G=\{\neg A_1,\cdots,\neg A_n\}$ 中搜索□的 SLD－消解证明。在分析寻找证明的搜索方法之前，我们先考虑会有什么结果。如果搜索证明的所有尝试都失败，PROLOG 回答“no”。如果找到了一个证明，PROLOG 给出一个回答替换(answer substitution)，即 G 中变元上的一个替换。事实上，如果编译器所找到的证明是带 mgu 序列 $\theta_0,\cdots,\theta_n$ 的 $\langle G_0,C_0\rangle,\cdots,\langle G_n,C_n\rangle$，那么它给出的回答替换是限制到 G 的变元上的$\theta=\theta_0\cdots\theta_n$。最重要的是，这些总是正确的回答替换，即$(A_1\wedge\cdots\wedge A_n)\theta$ 是 P 的一个逻辑后承。

定理 1.9(执行的可靠性) 如果 θ 是 $P\cup\{G\}$(带任一选择规则 R)的 SLD－反驳所给出的回答替换，那么 θ 是正确的，即$(A_1\wedge\cdots\wedge A_n)\theta$ 是 P 的一个逻辑后承。

证明 对 SLD－反驳的长度进行归纳。对于基本情形，即长度为 1 的反驳，$G=G_0=\{\neg A\}$ 和 $C_0=\{B\}$ 是带 θ 的单元素集，而 θ 是$\{A\}$和$\{B\}$的一个 mgu。由于 $B\in P$，所以 B 与它的替换实例 $B\theta$ 都是 P 的逻辑后承。由于 θ 是合一子，从而证得 $B\theta=A\theta$ 也是 P 的逻辑后承。现设 $G=\{\neg A_0,\cdots,\neg A_n\}$ 且 $P\cup\{G\}$ 有一个长度为 $n+1$ 的 SLD－反驳，该反驳从 $G_0=G$ 和$C_0=\{B,\neg B_0,\cdots,\neg B_m\}$在$\neg A_i$ 上用 mgu θ_0 所进行的消解开始。其消解式 $G_1=\{\neg A_0,\cdots,\neg A_{i-1},\neg B_0,\cdots,\neg B_m,\neg A_{i+1},\cdots,\neg A_n\}\theta_0$ 有一个从 P 出发的长度为 n 的带 mgu $\theta'=\theta_1\cdots\theta_n$ 的 SLD－反驳。因此，根据归纳假设，$G_1\theta_0\theta'$是 P 的逻辑后承。令 $\theta=\theta_0\theta'$。由于$C_0\in P$，$C_0\theta$ 也是 P 的一个逻辑后承。这里 $C_0\theta$ 等价于$(B_0\theta\wedge\cdots\wedge B_m\theta)\rightarrow B\theta$。所以根据命题逻辑，$\{\neg A_0\theta,\cdots,$

$\neg A_{i-1}\theta$, $\neg B\theta$, $\neg A_{i+1}\theta$, $\cdots$, $\neg A_n\theta\}$ 也是 P 的 一个结果。因为 $\theta=\theta_0\theta'$, θ 也合一 A_i 与 B, 即 $A_i\theta=B\theta$。因此，证得 $G\theta$ 是 P 的一个逻辑后承。 □

从某种意义上来说，SLD-消解提供的是所有的正确回答。

定理 1.10(执行的完全性) *令 R 为一个选择规则。如果 $P\models(A_1\wedge\cdots\wedge A_n)\sigma$ 且 $G=\{\neg A_1,\cdots,\neg A_n\}$，那么存在 $P\cup\{G\}$ 的一个带 R 的 SLD-反驳 $T=\langle\langle G_i, C_i\rangle\mid i\leqslant n\rangle$ 给出回答替换 θ，并且存在替换 ψ 使得 $G\sigma=G\theta\psi$。*

证明(对于恒定选择规则) 对 $P\cup\{G\sigma\}$ 的带 R 的 SLD-反驳的长度 n 作归纳，证明存在一个给出回答替换 θ 的 SLD-反驳，并且存在一个替换 ψ 使得在 G 中的变元上 $\sigma=\theta\psi$。选取一个替换 γ，将 $G\sigma$ 中的所有变元替换为新的常元(即不在 $P\cup\{G\}$ 中出现的常元)。由于 $P\models(A_1\wedge\cdots\wedge A_n)\sigma$，显然 $P\cup\{G\sigma\gamma\}\in\mathrm{UNSAT}$。令 $T=\langle\langle G_i, C_i\rangle\mid i<n\rangle$ 为 $P\cup\{G\sigma\gamma\}$ 的一个带 R 的基本 SLD-反驳。根据 R 的恒定性，还原替换 γ(即将常元替换为原来的变元)，可得 $P\cup\{G\sigma\}$ 的一个带 R 的 SLD-反驳 $T'=\langle\langle G_i', C_i'\rangle\mid i<n\rangle$，其中合一子 ψ_i 限制在 $G\sigma$ 中的变元上为恒等映射。根据习题 6，该基本反驳也可以提升 $P\cup\{G\}$ 的带 mgu 序列 θ_0, $\cdots$, θ_n 的 SLD-反驳

$$T''=\langle\langle G_i'',C_i''\rangle\mid i<n\rangle$$

设 R 从 G 中选取文字 $\neg A_i$。我们现在能应用归纳假设于

$$G'_1=\{\neg A_0\sigma,\cdots,\neg A_{i-1}\sigma,\neg B_{0,1}\psi_0,\cdots,\neg B_{0,m_0}\psi_0,\neg A_{i+1}\sigma,\cdots,\neg A_n\sigma\}$$

$$G''_1=\{\neg A_0\theta_0,\cdots,\neg A_{i-1}\theta_0,\neg B_{0,1}\theta_0,\cdots,\neg B_{0,m_0}\theta_0,\neg A_{i+1}\theta_0,\cdots,\neg A_n\theta_0\}$$

因为 $G_1'=G''_1\sigma\psi_0=G''_1\psi_0\sigma$(注意，$G\sigma\psi_0=G\sigma$ 并且由于 $\sigma\psi_0$ 合一 A_0 和 B_0，所以 $\theta_0\sigma\psi_0=\sigma\psi_0$)，所以有 $P\cup\{G''_1\}$ 的一个带 R 和 mgu 序列 θ_1', $\cdots$, θ_n'的 SLD-反驳，并且有一个 λ'使得在 G''_1 中的变元上 $\theta_1'\cdots\theta_n'\lambda'=\sigma$。如果 x 出现在 $G=G_0$ 中，但是 $x\theta_0$ 不出现在 G''_1 中，那么 $x\theta_0$ 不出现在θ_1', $\cdots$,θ_n'中。因为在 A_0 中的变元上 $\theta_0\sigma=\sigma$，所以对 A_0 中使得 $x\theta_0$ 不在 G''_1 中的每个变元 x，令$\lambda(x)=\sigma(x)$，可以将 λ'扩展为 λ。于是在 G 中的所有变元上有 $\theta_0\theta_1'\cdots\theta_n'\lambda=\sigma$,得证。(注意，对出现在 G 中的变元 y，$y\theta_0\sigma=y\sigma$。) □

为了对任意选择规则给出定理 1.8 和定理 1.10 的证明，我们现在证明独立性引理。下面先给出在一个 LD-反驳中改变其所选取的某个文字的基本方法。

引理 1.11(交换引理) *令 $G=G_0=\{\neg A_0, \cdots, \neg A_n\}$ 并且令$\langle G_0, C_0\rangle$, $\cdots$, $\langle G_k, C_k\rangle$ 为$P\cup\{G_0\}$ 的给出回答替换 $\psi=\psi_0\cdots\psi_k$ 的一个 LD-反驳。设 $A_j\psi_0\cdots\psi_{s-1}$ 为第 $s>0$ 步所消解的文字。则存在 $P\cup\{G_0\}$ 的给出回答替换 $\theta=\theta_0\cdots\theta_k$ 的一个 LD-反驳$\langle G_0, C_0'\rangle$, $\cdots$, $\langle G_k', C_k'\rangle$，其中在第 $s-1$ 步所消解的是 $A_j\theta_0\cdots\theta_{s-2}=A_j\psi_0\cdots\psi_{s-2}$，使得 $G\theta$ 是 $G\psi$ 的改名。*

证明 对$i\leqslant k$，令 $C_i=\{B_i, \neg B_{i,0}, \cdots, \neg B_{i,m_i}\}$。令 $G_{s-1}=\{\neg A_0', \neg A_1', \cdots, \neg A_t'\}$,其中 $A_j\psi_0\cdots\psi_{s-2}=A_{j'}'$，并设在第 $s-1$ 步消解 A_r'。因此

$$G_s=\{\neg A_0',\neg A'_1,\cdots,\neg A'_{r-1},\neg B_{s-1,0},\cdots,\neg B_{s-1,m_{s-1}},\neg A'_{r+1},\cdots,\neg A'_t\}\psi_{s-1}$$

并且 G_{s+1}是 G_s在用$\neg B_{s,0}$, $\cdots$, $\neg B_{s,m_s}$替换$\neg A_j=\neg A_{j'}'$之后再应用 ψ_s 所得的结果。(根据 LD-反驳的定义，$\neg B_{s,0}$, $\cdots$, $\neg B_{s,m_s}$ 中没有变元被 ψ_{s-1} 所作用。)我们知道 $A_{j'}'\psi_{s-1}\psi_s=B_s\psi_s=B_s\psi_{s-1}\psi_s$，从而能够合一 $A_{j'}'$和 B_s。令 ψ_{s-1}'为相应的 mgu，并令 λ 满足 $\psi_{s-1}\psi_s=\psi_{s-1}'\lambda$。现在用这个消解取代原反驳的第 $s-1$ 步。我们想证明在第 s 步可以用 mgu ψ_s'在文字 $A_r'\psi_{s-1}'$上消解 C_{s-1}。如果还能够证明这个消解的结果是 G_{s+1}的改名，那么这个反驳余下部分可以保持不

变，从而，不考虑命名方面的差异，得到所需要的结果。

首先注意到，在原来的反驳中，$A'_r\psi_{s-1}=B_{s-1}\psi_{s-1}$。结合这一事实与上面关于 λ 的关系得到如下等式序列：$A'_r\psi'_{s-1}\lambda=A'_r\psi_{s-1}\psi_s=B_{s-1}\psi_{s-1}\psi_s=B_{s-1}\psi'_{s-1}\lambda$。因此，$\lambda$ 合一 $A'_r\psi'_{s-1}$ 和 $B_{s-1}=B_{s-1}\psi'_{s-1}$（依惯例，$B_{s-1}$ 中没有变元被 ψ'_{s-1} 作用）。因而，正如所要求的那样，可以用 mgu ψ'_s 消解 $A'_r\psi'_{s-1}$。而且还得到一个 λ'，使得 $\lambda=\psi'_s\lambda'$。结合关于 λ 的那些等式，有 $\psi_{s-1}\psi_s=\psi'_{s-1}\psi'_s\lambda'$。如果现在可以证明存在一个替换 φ' 使得 $\psi'_{s-1}\psi'_s=\psi_{s-1}\psi_s\varphi'$，那么根据第二章第十一节习题 4，我们就证明了 $\psi'_{s-1}\psi'_s$ 是 $\psi_{s-1}\psi_s$ 的改名，从而完成证明。关于 φ' 存在的论证和 λ' 的情形相似。我们知道在原来的证明中 ψ_{s-1} 是 A'_r 和 B_{s-1} 的 mgu，从而 $A'_r\psi'_{s-1}\psi'_s=B_{s-1}\psi'_{s-1}\psi'_s$。因此存在 φ，使得 $\psi'_{s-1}\psi'_s=\psi_{s-1}\varphi$。然后注意到，$A'_{j'}\psi_{s-1}\varphi=A'_{j'}\psi'_{s-1}\psi'_s=B_s\psi'_s\psi'_{s-1}=B_s\psi_{s-1}\varphi$。因此，$\varphi$ 合一 $A'_{j'}\psi_{s-1}$ 和 $B_s\psi_{s-1}$。因为在原来的证明中 ψ_s 是这个合一子的 mgu，所以存在 φ' 使得 $\varphi=\psi_s\varphi'$。结合这一点和关于 φ 的第一个等式，有 $\psi'_{s-1}\psi'_s=\psi_{s-1}\psi_s\varphi'$，得证。

□

引理 1.12（独立性引理） *对于 $P\cup\{G\}$ 的给出回答替换 θ、长度为 n 的任何 LD－反驳和任何选择规则 R，存在 $P\cup\{G\}$ 的长度为 n 带 R 的 SLD－反驳给出回答替换 ψ，使得 $G\psi$ 是 $G\theta$ 的改名。*

证明 对给定的 LD－反驳的长度进行归纳证明。通常，对于长度为 1 的基本情形不需要证明。设有 $P\cup G$ 的一个长度为 $k+1$ 的给出回答替换 φ 的 LD－反驳。令 A_j 为 R 从 G 中选取的文字，并设它在给定的 LD－反驳的第 s 步被消解。应用引理 1.11 $s-1$ 次，可得 $P\cup\{G\}=P\cup\{G_0\}$ 的一个给出回答替换 $\varphi=\varphi_0\varphi_1\cdots\varphi_k$ 的 LD－反驳〈G_0, C_0〉，…，〈G_k, C_k〉，使得 A_j 是第 1 步所消解的文字且 φ 是 θ 通过 λ 的改名。现在应用归纳假设于 LD－反驳〈G_1, C_1〉，…，〈G_k, C_k〉，可得 $P\cup\{G_1\}$ 的一个带 R 的 LD－反驳给出回答替换 θ'，使得 $\varphi_1\cdots\varphi_k\lambda'=\theta'$，其中 λ' 是一个改名替换。现在可以把 $G=G_0$ 与 C_0（通过 mgu φ_0）在 A_j 上的消解放在这个反驳的前面，从而得到所需要的 $P\cup\{G\}$ 的给出回答替换 $\varphi_0\theta'$ 的带 R 的 SLD－反驳。注意到 $G_0\varphi_0\theta'=G_0\varphi_0\varphi_1\cdots\varphi_k\lambda'=G_0\varphi\lambda'=G_0\psi\lambda\lambda'$，据此完成归纳证明。我们现在已经找到所要求的 SLD－反驳和改名替换 $\lambda\lambda'$。 □

证明（定理 1.8 和定理 1.10） 现在，应用引理 1.12，由引理 1.4 立即可得定理 1.8。类似地，定理 1.10 可由引理 1.4 和前面对恒定选择规则所证明的特殊情形得到。 □

习题

1. 证明下面的引理：如果 G 是一个有序目标子句，C 是一个有序程序子句，G' 和 C' 分别是对应于 G 和 C 的无序子句（即对应的序列中各元素的并），并且目标子句 D' 是 G' 和 C' 的一个 LI－消解式，那么存在一个 LD－消解序列从 G 和 C 开始，并以 D' 所对应的有序目标子句 D 结束。
2. 应用习题 1 的结果，完成引理 1.4 带 * 的归纳证明。
3. 对于有序子句和 LD－消解，证明第二章引理 13.7。
4. 对于有序子句和 LD－消解，证明第二章引理 13.8。
5. 对于有序子句和 LD－消解，证明第二章推论 13.9。
6. 注意到，如果选择规则是恒定的，那么习题 3～5 的证明对于 SLD－消解也成立。
7. 令 S 为语言 $\mathcal{L}$ 中的子句集。S 的*成功集*（success set）是由 $\mathcal{L}$ 的谓词的所有基本实例 $A(t_1, \cdots, t_n)$ 构成的集合，使得 $S\cup\{\neg A(t_1, \cdots, t_n)\}$ 有一个消解反驳。证明：如果 P 是一个 PROLOG 程序，那么一个基本原子公式 $A(t_1, \cdots, t_n)$ 在 P 的成功集中，当且仅当它在 P 的所有厄布朗模型中都为真。
8. 设图 G（作为边的数据库）表示一个由所有事实“edge(n, m).”构成的列表，其中 n, m 是此图中任何有

一条边相连的一对顶点。(假设该图是无向图，从而“edge(n, m).”出现在数据库中，当且仅当“edge(m, n).”也出现在其中。)用一个 PROLOG 程序定义谓词“connected(X, Y)”，使得 connected(n, m)是该程序的逻辑后承，当且仅当存在一个顶点序列 $n=n_1$, n_2, …, $n_k=m$ 使得前后相继的每一对顶点 n_i, n_{i+1}之间有边相连。

第二节　执行：搜索与回溯

虽然 SLD -消解既可靠又完全，但是已有的 PROLOG 执行并不具备这两个性质。问题有两个：第一个问题，至少在理论上较小的一个问题，是合一算法的实现。正如我们前面所说的，已有的 PROLOG 执行在合一算法中省略了“出现检查”(occurs check)。因此，举例来说，PROLOG 的合一子相信 X 与 $f(X)$ 是可合一的。第二个问题是，PROLOG 的定理证明器仅在进行 SLD -消解时才作合一子所需要的替换。这两个问题合起来破坏了系统的可靠性。例如，给定程序：

```
test :- p(X,X).
p(X,f(X)).
```

和问题“?- test.”，PROLOG 将回答“yes.”。事情的经过是，定理证明器说要验证“test”，我们必须验证 $p(X, X)$。因为给定 $p(X, f(X))$，只需看 $p(X, X)$ 与 $p(X, f(X))$ 是否可以合一。合一子回答说它们可以合一。因为不需要其他信息(比如，是否需要实例化“test”)，所以证明器对我们的问题回答“yes”。因此，它所给出的答案就不是程序的逻辑后承——违反可靠性。

上述执行中的关键在于定理证明器没有实际执行合一子所默认的替换 $\{X/f(X)\}$。假如它执行该替换并试着给出替换结果，那么它就会陷入无穷循环当中。这种情况可用下面的程序说明：

```
test1(X) :- p(X,X).
p(X,f(X)).
```

如果现在问“?- test1(X).”，那么在 PROLOG 尝试给出 $X=f(f(f(\cdots$ 的“正确”回答替换时，所显示的是不断循环的结果。(点击 control-break 停止显示。)在搜索某个 SLD -消解时也会出现这种情况。考虑下面程序：

```
test2 :- p(X,X).
p(X,f(X)) :- p(X,X).
```

对于这个程序，唯一的问题是没有答案输出。

可是，这些问题在通常的程序中也会出现。当然这两个问题可以通过正确执行合一算法来消除。因为现在已有相当高效(即线性)的合一算法，所以这是一个合理的期待。不过就目前而言，人们已经可以相当轻松地解决第一个问题，这也为解决第二个问题奠定了基础。在编写程序时，让子句头部出现的变元也在其体部出现。对于在子句头但不在子句体出现的变元 X，在子句体上添加 $X=X$ 即可。这样，第一个程序变成

```
test :- p(X,X).
p(X,f(X)) :- X = X.
```

现在，当定理证明器要消解 $p(X, X)$ 和 $P(X, f(X))$ 并且合一子说它们可合一时，证明器要求给出这样的替换以便它可以用 $X=X$ 的替换结果取代目标子句中的 $p(X, X)$。由于这个替换是循环的，证明器将无法完成证明。于是，这个简单的编程技巧又恢复了 PROLOG

证明器的可靠性(当然，代价是增加了运行时间)。然而，完全性是另外一回事。

PROLOG 执行的不完全性来源于搜索 SLD -消解的方法。设给定一个程序 P 和一个目标子句 G。我们希望找到 $P\cup\{G\}$ 的一个由 G 开始的 SLD -消解反驳。在 SLD -消解的每一步 i，唯一要做的选择是确定用 P 中的哪一个子句去消解当前子句 G_i 中最左边的项。因此，可以把所有可能的 SLD -推导构成的空间表示成一棵树：根节点标记为 G，并且如果某个节点的标记为 G'，那么其所有后继的标记就是可以用于消解 G'最左边项的 P 中的所有子句。我们称这样的树为 P 和 G 的 SLD - 树。作为一个简单例子，考虑下面的程序 P_1：

$$p(X,X) :- q(X,Y),r(X,Z). \qquad (1)$$
$$p(X,X) :- s(X). \qquad (2)$$
$$q(b,a). \qquad (3)$$
$$q(a,a). \qquad (4)$$
$$q(X,Y) :- r(a,Y). \qquad (5)$$
$$r(b,Z). \qquad (6)$$
$$s(X) :- q(X,a). \qquad (7)$$

P_1 的以目标子句 $G=\{\neg p(X,X)\}$ 开头的 SLD -树如图 39 所示。我们在每个分支上标出了从 P_1 中选取用于消解的一个子句。习惯上，后继是按从左到右的次序排列的，与 P_1 中子句出现的次序相一致。对应着"yes"回答的*成功路径*以□结尾。在每个成功路径的末端，我们加上该路径所对应的(□的)证明所给出的回答替换。一个路径是*失败路径*，如果它以某个子句 G'结尾，而 P 中没有子句可以消解 G'最左边的项。

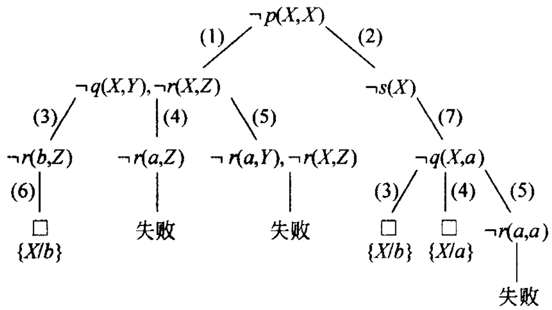


图 39

我们看到，在六条可能路径中，有三条失败路径、两条以正确替换$\{X/b\}$结尾的成功路径和一条以正确替换$\{X/a\}$结尾的成功路径。

PROLOG 定理证明器总是首先从 SLD -树最左边的路径开始寻找成功路径。也就是说，它尽可能用 P 最前面的子句去消解当前的 G。在图 39 中，它简单地沿着路径(1)，(3)，(6)找到正确的回答替换$\{X/b\}$。当定理证明器碰到一个*失败点*(failure point)(即不是□并且不能进行消解)时，它就*回溯*。回溯是指沿当前路径返回直到在某个节点处发现一个向右的路径。如果向右的路径不止一条，它就选择最左边的那条。定理证明器重复这样的回溯动作，直至找到一条成功路径。

在本节末尾给出程序 P_1(以及本节所列其他程序)运行过程的打印件，其中包括带"跟踪(tracing)"的运行过程。(跟踪是多数 PROLOG 执行所提供的一个功能，它显示在 SLD -树上

实际搜索的每一步。它是理解程序的控制流(flow of control)和调试程序的重要工具。注意，形如“_ 0nnn”的项仅仅是变元的名字。)

例如，如果从上面的程序 P_1 中去掉子句(3)得到程序 P_2，那么就有如图 40 所示的新的 SLD -树。

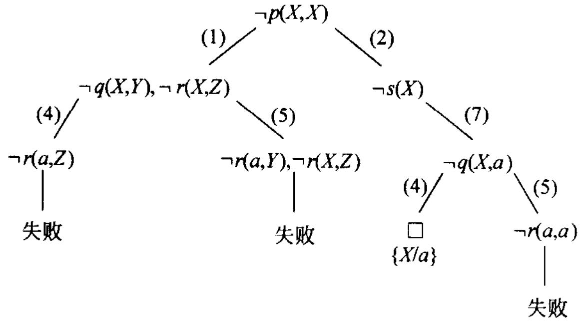


图 40

在这里，定理证明器首先尝试路径(1)，(4)，失败。然后，它回溯至 $\neg q(X, Y)$，$\neg r(X, Z)$并尝试(5)，失败。之后，它一路回溯到$\neg p(X, X)$并尝试(2)，(7)，(4)，成功地给出回答替换 $X=a$。

当要求 PROLOG 给出第二个答案时，它执行同样的回溯动作。因此，对于原来的程序 P_1 与目标$\{\neg p(X, X)\}$，即“? - p(X, X).”，我们得到的答案是“$X=b\rightarrow$”。如果现在通过输入“;”要求给出第二个答案，定理证明器就从上一次到达的成功的节点开始回溯，直到碰到一个有新路径的节点——这里是 $\neg q(X, Y)$，$\neg r(X, Z)$。然后，它尝试(4)失败，尝试(5)失败，然后回溯至$\neg p(X, X)$。现在它尝试(2)，(7)，(3)，再次给出答案 $X=b$。如果再一次输入“;”，它就从刚才的成功节点回溯至$\neg q(X, a)$，尝试(4)并给出答案 $X=a$。假如再一次要求给出其他的答案，PROLOG 将回溯到$\neg q(X, a)$，尝试剩下的路径(5)，失败并且报告说“no”——没有发现其他的成功节点并且没有其他的路径需要尝试。

对于第一次输入的问题，如果得到回答“no”，其情形是类似的。这意味着 SLD -树的所有路径都已经尝试过并且是失败的。根据一般完全性定理知，$P\cup\{G\}$ 是可满足的，并且不存在任何替换使得所问的问题是 P_1 的逻辑后承。根据执行的完全性定理知，当输入一串“;”询问之后最终得到回答“no”时，每个正确回答替换都是所给出的某个回答替换的实例。

这种搜索方法称为**深度优先搜索**(depth-first search)方法，因为它在沿着其他分支搜索之前，一直沿着一条路径尽可能深地往下走。相反，在如图 40 所示的树中沿着顺序$\neg p(X, X)$；$\neg q(X, Y)$；$\neg r(X, Z)$；$\neg s(X)$；$\neg r(b, Z)$；$\neg r(a, Z)$；$\neg r(a, Y)$；$\neg r(a, Z)$；$\neg q(X, a)$；□；失败；失败；□；□；$\neg r(a, a)$；失败搜索的方法，称为**广度优先搜索**(breadth-first search)。显然，还有许多混合的策略。在我们的例子中，深度优先搜索比广度优先搜索快得多(3 步对 9 步)。事实上，这是一个十分普遍的现象。深度优先搜索通常都比广度优先搜索快得多。显然，这就是 PROLOG 执行采用深度优先搜索的原因，然而代价也相当高昂——我们失去了 SLD -消解方法的完全性(与执行合一的方法是完全独立的)。

一般完全性定理保证，如果 $P\cup\{G\}\in$ UNSAT，则存在一个(必然是有穷的)由 G 开始的 SLD -反驳证明。如果有一个长度为 n 的反驳，那么应用广度优先搜索方法搜索至树的深度

n 时，即我们已经遍历了每个长度为 n 的可能路径时，就一定能找到这个证明。可是，深度优先搜索不能给出这样的保证。问题在于树的有些路径可以是无穷长的。深度优先搜索会让我们永远保持在这样的路径上，无法得到存在于另外一条路径上的简短证明。作为例子，考虑用下面的子句取代 P_1 中的(6)所得到的程序 P_3：

$$r(W,Z) :- r(b,Z). \qquad (6')$$

P_3 的以 $\neg p(X, X)$ 开头的 SLD-树如图 41 所示。

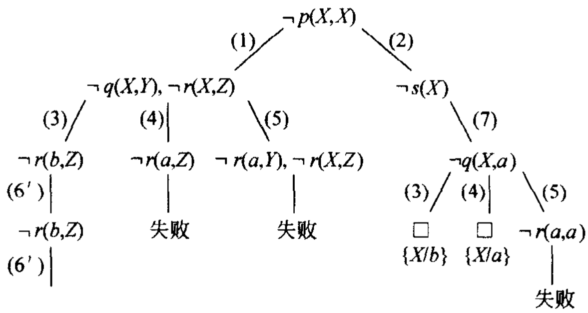


图 41

从图中可以看到，尽管沿着路径(2)，(7)，(3)和(2)，(7)，(4)都有 SLD-反驳，但是 PROLOG 所采用的深度优先搜索将不会发现它们。相反，它会沿着最左边的路径(1)，(3)，(6′)无休止地寻找，然后永远不停地使用子句(6′)。

现在可以看出，为什么子句的顺序在一个 PROLOG 程序的实际运行中起着关键作用。如果交换 P_3 中的子句(1)和(2)得到 P_4，定理证明器就会首先找到证明(2)，(7)，(3)，然后找到证明(2)，(7)，(4)。只有此后再次询问答案时，它才会陷入该树的无穷搜索部分。可是，重排子句的顺序固然是一个重要的编程想法，但不足以保证完全性。为了看清这一点，考虑程序 P_5：

$$\text{equivalent}(X,Y) :- \text{equivalent}(Y,X). \qquad (1)$$
$$\text{equivalent}(X,Z) :- \text{equivalent}(X,Y), \text{equivalent}(Y,Z). \qquad (2)$$
$$\text{equivalent}(a,b). \qquad (3)$$
$$\text{equivalent}(b,c). \qquad (4)$$

和目标子句 $G = \neg\text{equivalent}(c, a)$。可以看出，equivalent$(c, a)$ 是 P_5 的逻辑后承(习题 1)。问题是不管子句(1)和(2)在程序中的排序如何，深度优先搜索将总是尝试应用它们中的第一个，因为对于任何项 t_2 和 t_3，两个子句的头都可以与任何形如 equivalent(t_2, t_3) 的表达式合一。因此，定理证明器能够应用(1)或(2)，但是不能两个都应用。然而易知，如果从 P_5 中省掉这两个子句中的任何一个，其结果与 G 的合取是可满足的。因此，不管 P_5 中子句的顺序如何，没有深度优先搜索可以找到 $P \cup \{G\}$ 的 SLD-反驳。(注意，即使这里采用其他选择规则，深度优先的执行方式仍然找不到证明。)因此，深度优先搜索方法尽管效率高，本质上却是不完全的。在下一节，给出一个编程工具以最小化或者回避这些问题。在第四节中，简要地考虑(特定类型的)PROLOG 程序的一般终止问题，其中的程序采用标准选择规则和对 SLD-树的深度优先搜索方法。在第八节，证明 PROLOG 程序的一般终止问题是不可判定的。

习题

1. 找出 $P_5 \cup \{\neg\text{equivalent}(c, a)\}$ 的一个 SLD-反驳。

2. 证明：每个程序子句集 P 都有一个极小的(实际是最小的)厄布朗模型 M_P(即 P 的所有厄布朗模型的交还是 P 的厄布朗模型)，但是并非任何全称语句集 S 都有极小的厄布朗模型。

3. 考虑由下面三行构成的程序 P：

$$p(X,Z) :- q(X,Y), p(Y,Z). \qquad (1)$$
$$p(X,X). \qquad (2)$$
$$q(a,b). \qquad (3)$$

对于标准选择规则和总是选取子句中最右边文字的选择规则，分别画出 P 与目标“?-$p(X, Y)$.”的 SLD-树。

4. a)画出下面程序和目标的 SLD-反驳树，以表示对采用标准选择规则的 SLD-证明及其正确回答替换的所有构造尝试。

(1)$p(X, Y)$:-$s(X)$, $t(X)$.

(2)$p(X, Y)$:-$q(X, f(X))$, $r(X, f(Y))$.

(3)$q(b, Y)$.

(4)$q(a, Y)$:-$r(a, f(a))$.

(5)$r(a, f(a))$.

(6)$r(b, f(b))$.

(7)$s(a)$.

(8)$s(b)$.

(9)$t(a)$.

目标:?-$p(X, Y)$.

b)用 PROLOG 搜索与回溯等术语解释，当输入“?-$p(X, Y)$.”并且反复地用“;”询问更多答案时会发生什么。

5. 定义下列关于列表操作的 PROLOG 程序(列表的符号表示已在第二章第三节中介绍)。

a)second_element(X, Y)用于定义谓词“X 是列表 Y 的第二个元素”。

b)substitute_for_second_element(Y, $L1$, $L2$)用于定义谓词“$L2$ 是由 $L1$ 以 Y 替换其第二个元素而得”。

c)switch(L, M)用于定义谓词“M 是由 L 交换其第一个与最后一个元素而得”。

d)通过运行一些例子检验你的程序是否正确，至少在所有变元都被实例化之后：b 是不是[a, [b, c], d, e]或者[a, b, c]的第二个元素？[b, c]是不是它们的第二个元素？[a, [b, c], d]是用[a, [b, c], d]替换[a, b, c]的第二个元素的结果吗？[a, [a, [b, c], d], c]是吗？[a, c, b]是[a, b, c]交换其第一个与最后一个元素的结果吗？[c, b, a]是吗？

e)再试试有未被实例化的变元的例子：找出[a, [b, c], d, e]和[a, b, c]的第二个元素；用[a, [b, c], d]替换[a, b, c]的第二个元素，再用 b 替换所得列表的第二个元素；交换[a, b, c]的第一个与最后一个元素，再对所得的列表作同样的交换。现在尝试下面的一些例子，其中有的变元被实例化了，有的变元未被实例化：给出一个列表，其第二个元素为 b；给出一个列表，其第二个元素被 b 取代后可得[a, b, c]；给出一个列表，其第一个与最后一个元素交换后可得[a, b, c]。

f)根据上面的例子，关于你的程序可以得出哪些简单的全称断言呢？作为一个 PROLOG 问题，尝试询问是否用一个列表 X 的第二个元素替换其第二个元素所得的结果还是最初的 X。类似地，询问对 X 的第一个与最后一个元素相互交换两次所得的结果是否还是 X。试解释在输入这些问题后所得的输出。

6. 跟踪你的程序对习题 5 中例子的执行，并解释任何反常行为(也可能没有)。

在习题 7~8 中，我们用一元函数符号 s(意思是“后继”，successor of)来定义一个从常数 0 开始的计数器。因此，0 对应 0，$s(0)$对应 1，$s(s(0))$对应 2，一般地，$s^n(0)$(=$s\cdots s(0)$，其中函数 s 重复 n 次)对应 n。不要使用 PROLOG 内置的算术运算。

7. 编写一个程序，用这个计数器计算列表的长度。试试下面的两个例子：[a, b, c]，[a, b, c, [b, c], [d, c, e]]。当你尝试寻找一个长度为 3 的列表时会发生什么？

8. 在我们的语言中加上两个谓词符号"add(X, Y, Z)"和"multiply(X, Y, Z)"。

a)试证明：如果把"0"解释成自然数0并把"s"解释成后继函数，那么下面两行PROLOG程序就在如下的意义上定义了$\{0, s(0), s(s(0)), \cdots\}$所表示的自然数上的加法，即"add($s^n(0)$, $s^m(0)$, $s^r(0)$)"是该程序的一个逻辑后承，当且仅当 $n+m=r$：

$$\text{add}(X,0,X). \qquad (1)$$

$$\text{add}(X,s(Y),s(Z)) \text{ :- } \text{add}(X,Y,Z). \qquad (2)$$

b)编写一个类似的PROLOG程序用加法和后继定义乘法，使得"multiply($s^n(0)$, $s^m(0)$, $s^r(0)$)"是该程序的一个逻辑后承，当且仅当 $mn=r$。（提示：$x(y+1)=xy+x$。）

c)试证明："multiply($s^n(0)$, $s^m(0)$, $s^r(0)$)"事实上是该程序的一个逻辑后承，当且仅当$mn=r$。

9. 回忆在第二章第五节中定义的国际象棋棋盘上骑士的一步移动。不用任何内置谓词（即不用算术、cut和"not"），写出a)～c)所要求的程序。

a)编写一个程序，定义皇后的一步移动。（它可以纵向、横向或沿着对角线移动。）

b)编写一个程序，定义什么时候皇后不能从一个位置走到另一个位置。

c)编写一个程序，它可以找到一种办法在每一列上放置一个皇后，使得每个皇后都不能移动到其他皇后所在的方块内。

d)用c)中的程序找到那里所给出的国际象棋问题的两种解决办法。

10. 编写一个程序定义函数FLATTEN，除了最外面的括号之外，它消除列表中其他所有的方括号，即它给出由原子构成的列表，其中原子的顺序为所输入的列表的书写顺序。

11. 考虑下面的程序：

$$tc(X,Y) \text{ :- } r(X,Y).$$

$$tc(X,Y) \text{ :- } r(X,Z), tc(Z,Y).$$

事实上，$tc(a, b)$恰是这个程序和边数据库的逻辑后承，仅当(a, b)在第二章第七节习题11所定义的二元关系r的传递闭包之中。你如何将这个结论与那个习题的结果联系起来？（这个问题也与第一节习题8有关。）

下面的问题是第二章第五节习题7～8的继续，并且遵守那里关于家谱数据库的约定。数据库的打印件和一些警示语句包含在附录B中。这些问题还用到了习题7中所定义的计数器。再强调一下，假如不使用数据库的话，就直接写出程序定义给定谓词，并解释如何得到所需信息。这里，除了解释为什么你的答案在语义上正确之外，还要讨论它们作为PROLOG程序将会如何运行。

12. a)定义nthgrandfather(X, Y, $s^n(0)$)以表示X是Y的n代以前的一个祖先。（从$n=1$表示X是Y的父亲开始。）

b)用这个程序找出libni的祖父的曾祖父。

c)你能找出两个以上的这样的祖先吗？如果数据库没有问题的话，你能够做到这一点吗？

d)用这个程序找出levi的四个第三代曾孙(great-great-great-grandchildren)。

e)找出esau的三个孙子。

13. a)定义cousin(X, Y)，表示X与Y是通常意义下的第一代堂表亲。（译者注：见下面英文用法注释。）

b)找出tola的五个堂表亲。

c)定义secondcousin(X, Y)，表示X与Y是第二代堂表亲。

d)找出libni的六个第二代堂表亲。

回忆英文的用法：我的孩子与我兄弟的孩子是第一代堂表亲(first cousin)，我的孙子与他的孙子是第二代堂表亲(second cousin)，我的孩子与他的孙子是 第一代差一辈的堂表亲(first cousin once removed)，我的孩子与他的曾孙是第一代差两辈的堂表亲(first cousin twice removed)，我的孙子与他的曾孙是第二代差一辈的堂表亲(second cousin once removed)。

e)定义cousin(X, Y, $s^n(0)$, $s^m(0)$)，表示X是Y的第n代差m辈的堂表亲。

f)找出libni的七个第二代差一辈的堂表亲。你能否判断或者猜测并验证他们与d)中所列的人是什么关系？

g)找出libni的三个第三代差两辈的堂表亲。你能否预言，你的程序大概会如何找出他们与libni的关系

(即沿着什么样的路径找到这些实例)?

14. 在处理实际家谱数据库的时候,各种各样的反常情况会潜入你的结果中。考虑下列关于数据库的典型问题,它们可能影响到你的程序。

a)同一个数据可能被记录了两次。例如,事实 fatherof(abraham, isaac)在数据库中出现了两次。这会导致错误的答案吗?如果有影响的话,那么它会对程序运行造成什么影响呢?比如,考虑习题 12 中关于祖先或者习题 13 中关于第 n 代差 m 辈堂表亲的程序。

b)不同的人会有相同的名字,我们的数据库也是这样的,即对同名的人未加区分。(比如,看一看谁是 enoch 的父亲。)这会对关于祖先和堂表亲的程序造成什么影响?你能设计一个方法,识别数据库中的人,从而消除或者削弱这一问题的影响吗?(提示:考虑使用计数器。)再次考虑 enoch 的父亲,并看看你有没有消除 ancestor(X, X)的所有解。

c)同一个人可以有几个不同的名字。例如,我们知道在不同的《圣经》篇章中 esau 就是 seir。这会对关于祖先和堂表兄的程序造成什么影响?(不编辑数据文件)你如何纠正这种情况?不编辑数据库,你能在数据库中添加一些规则解决这个问题吗?(以找出数据库中 lotan 的祖父和 hori 的堂表亲为例,试一试你的解决办法。)

程序 $P_1 \sim P_4$ 的运行

程序 P_1.

```
?- listing.
p(A,A) :-
   q(A,B),
   r(A,C).
p(A,A) :-
   s(A).
q(b,a).
q(a,a).
q(A,B) :-
   r(a,B).
r(b,A).
s(A) :-
   q(A,a).
yes
?- p(X,X),
X = b →;
X = b →;
X = a →;
no
?- trace.
yes
?- leash(full).
yes
?- spy(p/2).
yes
?- p(X,X).
  ** (0) CALL: p(_0085,_0085)?>
  (1) CALL: q(_0085,_0255)?>
  (1) EXIT: q(b,a)?>
  (2) CALL: r(b,_0261)?>
  (2) EXIT: r(b,_0261)?>
  ** (0) EXIT: p(b,b)?>
X = b →;
  ** (0) REDO: p(b,b)?>
  (2) REDO: r(b,_0262)?>
  (2) FAIL: r(b,_0261)?>
  (1) REDO: q(b,a)?>
  (1) EXIT: q(a,a)?>
  (3) CALL: r(a,_0261)?>
  (3) FAIL: r(a,_0261)?>
  (1) REDO: q(a,a)?>
  (4) CALL: r(a,_0255)?>
  (4) FAIL: r(a,_0255)?>
  (1) FAIL: q(_0085,_0255)?>
  (5) CALL: s(_0085)?>
  (6) CALL: q(_0085,a)?>
  (6) EXIT: q(b,a)?>
  (5) EXIT: s(b)?>
  ** (0) EXIT: p(b,b)?>
X = b →;
  ** (0) REDO: p(b,b)?
  (5) REDO: s(b)?>
  (6) REDO: q(b,a)?>
  (6) EXIT: q(a,a)?>
  (5) EXIT: s(a)?>
  ** (0) EXIT: p(a,a)?>
X = a →;
  ** (0) REDO: p(a,a)?>
  (5) REDO: s(a)?>
  (6) REDO: q(a,a)?>
  (7) CALL: r(a,a)?>
  (7) FAIL: r(a,a)?>
  (6) FAIL: q(_0085,a)?>
  (5) FAIL: s(_0085)?>
  ** (0) FAIL: p(_0085,_0085)?>
no
```

程序 P_2.

```
?- listing.
p(A,A) :-
   q(A,B),
   r(A,C).
p(A,A) :-
   s(A).
q(a,a).
q(A,B) :-
   r(a,B).
r(b,A).
s(A) :-
   q(A,a).
yes
?- p(X,X).
X = a →;
no
?- trace.
yes
?- leash (full).
yes
?- spy(p/2).
yes
?- p(X,X).
  ** (0) CALL: p(_005D,_005D)?>
  (1) CALL: q(_005D,_022D)?>
  (1) EXIT: q(a,a)?>
  (2) CALL: r(a,_0239)?>
  (2) FAIL: r(a,_0239)?>
  (1) REDO: q(a,a)?>
  (3) CALL: r(a,_022D)?>
  (3) FAIL: r(a,_022D)?>
  (1) FAIL: q(_005D,_022D)?>
  (4) CALL: s(_005D)?>
  (5) CALL: Q(_005D,a)?>
  (5) EXIT: q(a,a)?>
  (4) EXIT: s(a)?>
  ** (0) EXIT p(a,a)?>
X = a →;
  ** (0) REDO: p(a,a)?>
  (4) REDO: s(a)?>
  (5) REDO: q(a,a)?>
  (6) CALL: r(a,a)?>
  (6) FAIL: r(a,a)?>
  (5) FAIL: q(_005D,a)?>
  (4) FAIL: s(_005D)?>
  ** (0) FAIL: p(_005D,_005D)?>
no
```

程序 P_3.

```
?- listing.
p(A,A) :-
   q(A,B),
   r(A,C).
p(A,A) :-
   s(A).
q(b,a).
q(a,a).
q(A,B) :-
   r(a,B).
r(A,B) :-
   r(b,B).
s(A) :-
   q(A,a).

yes
?- p(X,X).
?- trace.
yes
?- leash(full).
yes
?- spy(p/2).
yes
?- p(X,X).
  ** (0) CALL: p(_005D,_005D)?>
  (1) CALL: q(_005D,_022D)?>
  (1) EXIT: q(b,a)?>
  (2) CALL: r(b,_0239)?>
  (3) CALL: r(b,_0239)?>
  (4) CALL: r(b,_0239)?>
  (5) CALL: r(b,_0239)?>
```

程序 P_4.

```
?- listing.
p(A,A) :-
   s(A).
p(A,A) :-
   q(A,B),
   r(A,C).
q(b,a).
q(a,a).
q(A,B) :-
   r(a,B).
?- p(X,X).
  ** (0) CALL: p(_005D,_005D)?>
  (1) CALL: s(_005D)?>
  (2) CALL: q(_005D,a)?>
  (2) EXIT: q(b,a)?>
  (1) EXIT: s(b)?>
  ** (0) EXIT: p(b,b)?>
X = b →;
```

```
r(A,B) :-
   r(b,B).
s(A) :-
   q(A,a).

yes
?-  p(X,X).
X = b →;
X = a →;
?-  trace.
yes
?-  leash(full).
yes
?-  spy(p/2).
yes
   ** (0) REDO: p(b,b)?>
   (1) REDO: s(b)?>
   (2) REDO: q(b,a)?>
   (2) EXIT: q(a,a)?>
   (1) EXIT: s(a)?>
   ** (0) EXIT: p(a,a)?>
X = a →;
   ** (0) REDO: p(a,a)?>
   (1) REDO: s(a)?>
   (2) REDO: q(a,a)?>
   (3) CALL: r(a,a)?>
   (4) CALL: r(b,a)?>
   (5) CALL: r(b,a)?>
```

第三节 执行的控制：cut

我们已经看到，即使是语义正确的 PROLOG 程序，其成功运行也依赖着其执行的方方面面。至今，我们对于执行过程的唯一控制是给程序的子句排序。例如，我们知道递归的基本情形应当总位于归纳的基本情形之前。(为什么?)类似地，关于谓词的事实一般应当位于断言性规则的前面。但是这些启发性做法走不了多远。通常我们希望对 SLD -树的实际搜索做更详细的控制。有时，这“仅仅”是为了提高效率。而在另外一些时候，似乎根本没有别的办法可以使程序运行。本节考虑这样的一个内部控制工具——cut。

从语法上来说，cut(记作“!”)表面上不过是又一个文字。因此有：

$$p :- q_1, q_2, !, q_3, q_4.$$

但是，它没有任何(指定的)语义。其功能是改变程序的运行。在搜索 SLD -树时调用上面的子句，子目标 q_1，q_2,!，q_3，q_4 像通常那样被添加到当前目标子句的开头。和以前一样，我们设法依次满足 q_1 和 q_2。如果成功了就跳过那个 cut 并尝试满足 q_3 和 q_4。如果满足了 q_3 和 q_4，一切照常继续进行，就和没有这个 cut 一样。然而，假如失败并且因此被回溯至那个 cut，我们就当是 p 失败了并且“深度回溯(deep backtracking)”至 SLD -树中恰好在 p 上面的那个节点，称为父目标，然后尝试该节点右边的那个分支。(同往常一样，如果没有这样的分支，当前搜索失败。)

例 3.1 考虑下面的程序：

$t :- p, r.$	(1)
$t :- s.$	(2)
$p :- q_1, q_2, !, q_3, q_4.$	(3)
$p :- u, v.$	(4)
$q_1.$	(5)
$q_2.$	(6)
$s.$	(7)
$u.$	(8)

对于目标 $\{\neg t\}$ 我们得到如图 42 所示的 SLD -树。

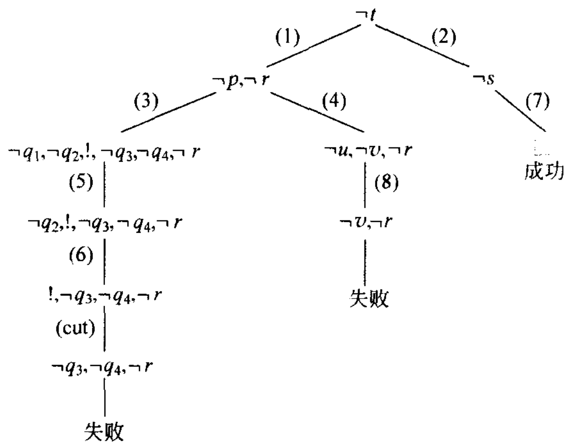


图 42

和往常一样，当进行搜索时，我们沿着最左边分支向下行走，穿越那个 cut 直到遇上失败节点。(注意，“!”根据定义成功地经历了第一次穿越；这时我们就把它删除。)然而，当我们碰到最左边路径末端的失败节点时，控制就被传递到标记为$\neg s$的节点上，由此我们立刻成功。这里的要点是，一旦 cut 后面的目标子句$\neg q_3, \neg q_4, \neg r$失败，不是回到那个含 cut 的子句的父节点($\neg p$, $\neg r$)，而是回到($\neg p$, $\neg r$) 的父节点 ($\neg t$)。然后尝试右边的那个分支(通过$\neg s$)继续搜索。

在这个例子中，通过使用 cut 切除 SLD -树的失败分支仅节省了程序运行的时间。cut 这样的使用当然是无害的(或者安全的(safe))。然而要保证所用的 cut 都是安全的让程序编写变得困难。一般地，cut 会导致不完全性和不可靠性。显然，如果 cut 剪掉一条成功路径的话，那么就会丢失一些正确的回答替换。更坏的是，它也许会剪掉所有的成功路径。这样我们就会走上一条无穷路径——如同对深度优先搜索所做的分析一样，这表明该搜索系统是不完全的。最后它也可能剪掉所有的成功路径和无穷路径。在这种情况下，当 $P\cup\{G\}$ 实际是不可满足的时，PROLOG 将回答“no”。这样，cut 导致了程序的不可靠性，因为回答“no”意味着 $P\cup\{G\}$ 是可满足的。

不管怎么说，只要在使用的时候足够谨慎，cut 还是很有用的。PROLOG 执行还有其他控制搜索的工具。其中一个称作 snip(剪切)。它是 cut 的一种限制形式。例如，遇到子句 p:-q_1, q_2, [!q_3, q_4!], q_5 时，搜索通常穿过裁剪片断(q_4)，即穿过两个感叹号之间的子句。回溯本应把你送至裁剪片断的右端，相反它跳过裁剪片断返回到 q_2。尽管这偶尔是个便利的工具，但是在习题中看到，snip 总是可以用 cut 来代替的。一般地，我们要告诫读者——当心，cut 会导致意想不到的后果。至少，应当避免使用破坏程序本来语义的 cut。让我们再来简要地看一下 cut 的一个重要应用——定义否定谓词——作为其他论题的导入。

尽管“not”是 PROLOG 的内部谓词(built-in predicate)，我们还是用 cut 定义它，从而弄清它的真正含义。它在 PROLOG 中的意思不同于通常逻辑否定。“not(P)”的意思是有一个 P 失败的证明(即，P 是不可证明的)。因此 PROLOG 对于目标“not(A)”回答“yes”，当且仅当它对目标 A 回答“no”。我们可以用 cut 取代“not”，只需加入“not(A)”的一个定义：

```
not(A) :- A, !, fail.
not(A).
```

这里我们看到，如果调用 not(A)，PROLOG 就运行其定义中的第一个子句并调用 A。如果 A 成立，我们越过 cut 并抵达“fail”。“fail”是一个总是不成立的内部谓词(可以用任何总不成立的子句代替)。因此，如果 A 成立，“not(A)”就不成立。另一方面，如果 A 不成立，我们就会用到定义“not(A)”的第二个子句，从而“not(A)”成立。

事实上，事情也可以反过来看。cut 很多的应用能够并且应当用“not”来代替。我们说“应该”是因为“not”在 PROLOG 的程序中可以有明确的含义(尽管不是经典的那个)，而说明 cut 却比较困难。在第六节中从理论上探讨 PROLOG 使用“not”的语义根据。为此，我们首先在第五节中从处理相等关系开始。

习题

1. 第二节习题 4 中在子句(2)的两个字面值间加入一个 cut(!)，对关于 b)的回答有什么影响吗？
回忆第二章例 3.15 中所介绍的列表符号。考虑下面连接列表的程序 APPEND：
(a1) a([ ], Y, Y).
(a2) a([X | Y], Z, [X | W]):- a(Y, Z, W).
2. 把这个程序修改成下面的 APPEND'有什么好处呢？
(a1) a([ ], Y, Y):- ! .
(a2) a([X | Y], Z, [X | W]):- a(Y, Z, W).
考虑这样的情形，有两个列表 x 和 y 并且希望得出将一个放在另一个之前的结果，也就是考虑形如?- a([x, y, z], [u, v], Z)的目标。再考虑形如?- a([x, y, z], V, W)的目标。
3. 当考虑形如?- a(X, Y, [x, y, z])的目标时(与 APPEND 相比)在 APPEND'的运行中会出现什么问题？考虑当你想要得到两个以上的回答替换时会发生什么？
4. 这个问题有一点宽泛并且需要用到第二章第五节的习题 7 ~ 8 和第三章第二节的习题 12 ~ 14 中所给出的数据库。你能用 cut 和数据库中子句的排序来缓解你以前在祖父、叔伯或者堂表亲等家庭关系的程序中所遇到的问题吗？假设数据库中的子句反映了出生的年代次序。你也许可以考虑同时改写你的程序与再写一个(使用 cut 的)新程序以修改数据库防止出现类似这样的结果：某人是他自己的祖父。

第四节 PROLOG 程序终止的条件

程序分析的一个重要问题是判断程序何时终止。当然，这个问题一般是不可判定的(定理 8.9和推论 8.10)，但是对于某些解决实际问题的程序，这个问题是可以解决的。本节给出一个分析方法，可以用来抽象地刻画这些程序的终止特征，它们采用总选取最左边的文字的标准选择规则。这里给出的是 Apt 和 Pedreschi[1991，5.4]的研究结果。他们改造了 Bezem[1989，5.4]中给出的方法，在选择规则固定为标准选择规则的情况下，该方法可以刻画给定程序和目标的所有 LD -证明的终止特征。

在本节其余部分，P 是某语言$\mathcal{L}$中一个 PROLOG 程序而 G 是$\mathcal{L}$中的一个目标。所有子句和消解都是有序的。令 P'与 G'分别表示 P 与 G 的基本实例。SLD -证明都采用标准选择规则。我们希望刻画的程序基本性质由下面的定义给出：

定义 4.1 P 对于目标 G 是**左终止的**(left-terminating)，如果 $P \cup \{G\}$ 的所有由 G 开始的 LD -证明都是有穷的。P 是**左终止的**，如果它对于所有基本目标 G 都是左终止的。

注意，如果 P 对于 G 是左终止的，那么 P 在 PROLOG 中的标准执行(使用最左选择和深度优先搜索)将在目标 G 上终止。其实，当所有以 G 开始的 LD -证明都是有穷的，那么不管使用什么选择规则，程序的运行都会终止。因此，如果能对给定的目标子句证明左终止性，

那么就可以确信我们的程序在标准运行中终止。

在关于推理终止的几乎所有证明中，基本的策略有两部分。首先，给子句或者证明小心地定义一个良序。其次，证明每一步推导(这里就是消解)都会沿着这个良序作一次下降。一旦到达这一步，显然可以看出所有的证明都会终止，因为证明的每一步都代表着良序中的一次下降。对于良序的描述我们从层映射的基本概念开始。

定义 4.2

(i)P 的**层映射**(level mapping)是从$\mathcal{L}$的原子语句(正的基本文字)到$\mathcal{N}$的一个函数 f。我们把该函数在文字 A 上的值记为 $|A|_f$，称为 A 的 **f 层次**。若函数 f 能从文中看出可以省去。

(ii)如果$\mathcal{M}$是$\mathcal{L}$的一个结构并且$\vec{A}=A_1$，…，A_n 是一个原子语句序列，令$\mathcal{M}(\vec{A})$为使得$\mathcal{M}\nvDash A_i$ 的最小的 $i\leqslant n$，如果这样的 i 存在的话，否则令它为 n 本身。如果 $G=\{\neg A_1$，…，$\neg A_n\}$，那么把$\mathcal{M}(A_1$，…，$A_n)$记为$\mathcal{M}(G)$。

(iii)**P 关于层映射 f 与 P 的模型$\mathcal{M}$是可接受的**(acceptable with respect to a level mapping f and a model $\mathcal{M}$ of P)，如果对于 P' 中每个 $B:-\vec{A}$，即 P 中子句的每个基本实例和每个 $i\leqslant\mathcal{M}(\vec{A})$，有 $|B|>|A_i|$。P 是**可接受的**(acceptable)，如果它关于某个层映射和模型是可接受的。

为掌握上面这个定义的思想，首先只考虑层映射不考虑模型。显然，如果 P 是可接受的，那么目标子句 $G=\{\neg A_1$，…，$\neg A_n\}$与 P 中子句进行的每一个基本消解其结果的层次低于 G。这个修正的可接受性概念对应于要求 P 的所有从 G 开始的 LD－证明都有穷这个条件。(见习题 1～2。)对 SLD－证明的这一限制可以通过选取适当的模型$\mathcal{M}$将文字 A_i 限制在 $i\leqslant\mathcal{M}(G)$范围之内来实现。其基本想法是，如果没有$\{\neg A_i\}$的 SLD－反驳，就没有越过 A_i 的由 G 开始的 SLD－反驳。因此，在分析由 G 开始的 SLD－树时，没有必要考虑后面的文字。另一方面，如果没有$\{\neg A_i\}$的 SLD－反驳，则存在一个模型$\mathcal{M}$，使得 A_i 在其中不成立。因此，正确选取的模型将在此处切断层映射，不再考虑后面的文字。

在开始我们的分析之前，根据给定的 P 的层映射 f 和模型$\mathcal{M}$，定义子句所需要的良序。我们必须先考虑基本目标子句，然后考虑其他非基本目标子句。

定义 4.3

(i)令 P 对于 f 和$\mathcal{M}$是可接受的。通过令 $|G_i|=\langle|A_{i,j_1}|$，…，$|A_{i,j_k}|\rangle$，将 f 扩张至基本目标子句 $G_i=\{\neg A_{i,1}$，…，$\neg A_{i,n_i}\}$上，其中文字 $A_{i,1}$，…，$A_{i,\mathcal{M}(G_i)}$按其层次递减顺序排列(不必是严格的递减)。我们以字典序来排列它们，(形式上，要使得 f 的这个扩张符合其字面意义，我们可以将每个 $n\in\mathcal{N}$视为单元素序列$\langle n\rangle$。)注意，根据第一章第一节习题 8(b)有穷序列上的字典序是良序。

(ii)目标子句 G 是(关于 f 和$\mathcal{M}$)**有界的**(bounded)，如果$\{|G^*|:G^*\in G'\}$中存在最大元。(注意 G'是 G 的基本实例集。)如果 G 是有界的，我们将其最大元记为 $|G|$。(注意，由于 n 与$\langle n\rangle$等同，当 G 是基本目标子句时，这个定义与(i)中定义一致。)

使用有界一词的合理性见下面的引理。

引理 4.4 $G=\{\neg A_1$，…，$\neg A_n\}$是(关于 f 和$\mathcal{M}$)有界的，当且仅当存在一个序列 τ，使得对于任何 $G^*\in G'$，有 $|G^*|\leqslant\tau$。

证明 易知必要条件成立。设存在引理所描述的序列 τ，令数 t 大于 τ 中每一个元素。

如果 $G^* \in G'$，那么在序列 $|G^*|$ 中至多存在 n 个元素。根据所选取的 τ 和 t，这些元素中的每一个都小于 t。因此，只有有穷多个形如 $|G^*|$ 的序列，使得 $G^* \in G'$。由于在任何序中有穷集必有极大元，引理得证。 □

下面的任务是证明：任何从有界目标 G 开始的可接受程序一定终止。因为目标子句上所导出的序是良序，只需证明用 P 中子句所作的消解降低了目标子句的层次。我们从基本消解开始。

引理 4.5 令 P 关于层映射 f 与模型 $\mathcal{M}$ 是可接受的。如果 $G=\{\neg A_1, \cdots, \neg A_n\}$ 是一个基本目标，而 $H=\{\neg B_1, \cdots, \neg B_m, \neg A_2, \cdots, \neg A_n\}$ 是 G 与 P' 中某个基本子句 $C=\{B, \neg B_1, \cdots, \neg B_m\}$ 的 SLD-消解式，那么 $|H| < |G|$。

证明 分两种情形讨论。首先，设 $\mathcal{M}(G)=1$，即 $\mathcal{M} \nvDash A_1$，从而根据定义有 $|G| = \langle |A_1| \rangle$。由于我们对 G 和 C 进行了消解，所以 $B=A_1$，并且根据假设得知，A_1 在 $\mathcal{M}$ 中不成立。因为 $C \in P'$，C 一定在 P 的模型 $\mathcal{M}$ 中成立。因此，一定有某个 $i \leqslant m$，使得 B_i 在 $\mathcal{M}$ 中不成立，从而根据定义有 $\mathcal{M}(H) \leqslant m$。因为根据可接受性，对任何 $i \leqslant m$，有 $|B_i| < |A_1|$，故根据所定义的序得 $|H| < |G| = |A_1|$。

其次，设 $\mathcal{M}(G) > 1$。此时，H 和 G 在模型 $\mathcal{M}$ 中在同一个文字上取得第一次成功，即 $\mathcal{M}(H) = \mathcal{M}(G) + m - 1$。因此，除了 $|A_1|$ 被集合 $\{|B_i|: 1 \leqslant i \leqslant m\}$ 取代之外，序列 $|H|$ 和 $|G|$ 具有相同元素。因为对每一个 i，$|B_i| < |A_1|$，根据层映射向子句扩张的定义以及序列上序的定义可知 $|H| < |G|$。 □

现在对有界目标子句证明我们的引理。

引理 4.6 令 P 关于层映射 f 与模型 $\mathcal{M}$ 是可接受的。如果 $G=\{\neg A_1, \cdots, \neg A_n\}$ 是一个（关于 f 和 $\mathcal{M}$）有界目标且 $H=\{\neg B_1, \cdots, \neg B_m, \neg A_2, \cdots, \neg A_n\}\theta$ 是 G 与 P 中某个子句 $C=\{B, \neg B_1, \cdots, \neg B_m\}$ 的 SLD-消解式，那么 H 有界且 $|H| < |G|$。

证明 考虑 H 的任何基本实例 $H\gamma$。因为在必要的时候可以扩张 γ，所以可设 $\theta\gamma$ 也把 B 化为基本实例。那么 $H\gamma$ 是 $G\theta\gamma$ 和 $C\theta\gamma \in P'$ 的 SLD-消解式，从而由引理 4.5 知，$|H\gamma| < |G\theta\gamma|$。因为 G 有界，$|G\theta\gamma| \leqslant |G|$。因为 $H\gamma$ 是 H 的一个任意的基本实例，由引理 4.4 知，H 有界。如果选取 γ，使得 $|H\gamma| = |H|$，即得所要证明的 $|H| < |G|$。 □

定理 4.7 如果 P 是可接受的而 G（相对于任何表明 P 是可接受的层映射与模型来说）是有界目标子句，那么 $P \cup \{G\}$ 的每个由 G 开始的 SLD-证明是有穷的。

证明 考虑 $P \cup \{G\}$ 的任何由 $G=G_0$ 开始的 SLD-证明。每个后继的消解产生一个新目标子句 G_n。根据引理 4.6，序列 $|G_n|$ 是严格递减的。因为目标子句上的序是良序，所以消解序列一定是有穷的。 □

推论 4.8 每个可接受程序都是左终止的。

证明 因为根据定义每个基本目标是有界的，由上面的定理可得该推论。 □

为了刻画左终止程序的特征，需要证明推论 4.8 的逆。还要证明定理 4.7 某种形式的逆以处理非基本目标。我们从层映射的各个组成部分开始讨论。

定义 4.9 如果 $P \cup \{G\}$ 的由 G 开始的 SLD-树是有穷的，那么 $N(G)$ 表示其节点数；否则，$N(G)$ 无定义。

定理 4.10 如果 P 是左终止的，那么存在一个层映射 f 和一个模型 $\mathcal{M}$，使得

(i) P 关于 f 和 $\mathcal{M}$ 是可接受的且

(ii)对任何目标子句 G，G 关于 f 和 $\mathcal{M}$ 有界，当且仅当 $P\cup\{G\}$ 的每个由 G 开始的 SLD-证明是有穷的。

证明 定义所要求的层映射 f 和模型 $\mathcal{M}$ 如下。对于 $\mathcal{L}$ 中每个原子语句 A，令 $|A| = N(\{\neg A\})$ 并且要求 $\mathcal{M}\vDash A \Leftrightarrow$ 存在 $P\cup\{\neg A\}$ 的一个 SLD-反驳。注意到，因为我们设 P 是左终止的，所以 f 的定义合理。此外，由 SLD-消解的完全性可知，任何原子语句 A 在 $\mathcal{M}$ 中为真，当且仅当它是 P 的逻辑后承。现在证明 f 和 $\mathcal{M}$ 具备所要求的性质。

(i)考虑 P' 中任何形如 $A:-B_1, \cdots, B_m$ 的子句 C。令 n 表示数 $\mathcal{M}(B_1, \cdots, B_m)$。存在 $P\cup\{\neg A\}$ 的从 $\{\neg A\}$ 开始的 SLD-证明和一个与 C 的消解。根据 $\mathcal{M}$ 的定义，对于每一个 $i<n$，存在 $P\cup\{\neg B_i\}$ 的一个从 $\neg B_i$ 开始的 SLD-反驳。每一个这样的反驳的 SLD-搜索树显然是 $\{\neg A\}$ 的 SLD-树的子树。因为对于 $i<n$ 的每个搜索都成功，所以把 $\neg B_n$ 的 SLD-树接在每个成功搜索到 $\neg B_i$ 反驳的路径的末端。因此对于每个 $i\leqslant n$，$\{\neg A\}$ 的 SLD-树包含 $\{\neg B_i\}$ 的 SLD-树。于是根据 f 的定义可得，对于每个 $i\leqslant n$，$|A|>|B_i|$。

(ii)令 G 为一个有界目标子句。为了得出矛盾，假设有一个从 $G=G_0$ 开始的不终止的 SLD-证明 $\langle G_0, C_0\rangle$，$\langle G_1, C_1\rangle$，…。对于任何 n，我们可以(从 $\langle G_n, C_n\rangle$ 开始)找到一个替换 θ，将 $\langle G_0, C_0\rangle$，$\langle G_1, C_1\rangle$，…，$\langle G_n, C_n\rangle$ 中的所有子句化为基本实例。这就给出一个长度为 n 的由 G 的基本实例 $G\theta$ 开始的 SLD-证明。由于 n 是任意的，这与假设 G 有界矛盾。

最后，设每个由 G 开始的 SLD-证明终止。注意到，从任何目标子句开始的 SLD-树的分支是有穷的：每个节点的每个直接后继对应着有穷程序 P 中的一个子句。因此，根据库尼西引理(第一章定理 1.4)G 的 SLD-树是有穷的。设它有 n 个节点。又因为每个 SLD-树其分支数至多为 P 中子句的个数，若存在节点任意多从 G 的基本实例开始的 SLD-树，那么其深度必定是任意大的。因此，如果 G 不是有界的，就存在长度为 $n+1$ 从 G 的基本实例开始的 SLD-证明。那么用 SLD-证明的提升引理(第一节习题 6)可以将这个证明提升为长度为 $n+1$ 由 G 开始的 SLD-证明。由于这个证明一定是由 G 开始的 SLD-树的一条路径，故得到所需要的矛盾。 □

推论 4.11 *P 是左终止的，当且仅当它是可接受的。*

作为例子，我们证明下面的程序 PERM 是左终止的。PERM 可以判断一个列表是否是另一个列表的置换。我们的程序语言包括第二章例 3.15 中表示空列表的常元[ ]和二元列表连接函数“.”。我们也使用那里所介绍其他符号，例如，用$[a]|[b, c, d]$表示$[a, b, c, d]$。该语言的基本项集合，即它的厄布朗域 H，由常量[ ]在列表连接运算下的闭包构成。程序 PERM 包括将一个列表连接到另外一个上面的程序 APPEND(由下面两个子句(a1)和(a2)组成)和两个应用 APPEND 定义 PERM 的子句(p1)和(p2)：

(a1) $a([\ ], Y, Y)$.

(a2) $a([X|Y], Z, [X|W]):-a(Y, Z, W)$.

(p1) $p([\ ], [\ ])$.

(p2) $p(X, [Y, Z]):-a(W, [Y|V], X), a(W, V, U), p(U, Z)$.

这里，不是所有 PERM 的由基本目标开始的 LD-证明都是有穷的(习题 3)，但我们证明 PERM 是可接受的，从而所有从基本目标开始的 SLD-证明将终止。

定理 4.12 PERM *是可接受的(从而是左终止的)*。

证明 首先，对厄布朗域 H 中任何列表 x，令 $|x|$ 为 x 的长度。因此举例来说，对任何 $y, v \in H$，$|[y|v]| = |v| + 1$。定义一个层映射如下：$|p(x, y)| = |x| + |y| + 1$ 且 $|a(x, y, z)| = \min\{|x|, |z|\}$。作为我们的模型，我们可以取 p 和 a 在 H 上预期的解释。与其验证该程序的语义正确性，不如构造一个模型更为容易，只要这个模型足以避免不必要的消解。定义以 H 为全域的模型 $\mathcal{M}$ 如下：对任何 $x, y \in H$，令 $p(x, y)$ 成立，且 $a(x, y, z)$ 成立，当且仅当 $|x| + |y| = |z|$。根据定义，显然 $\mathcal{M}$ 是 PERM 的模型。证明：对于我们所选取的层映射和模型，PERM 是可接受的。

我们只需验证子句(a2)和(p2)的基本实例。对于(a2)注意到，根据定义，对于任何 $x, y \in H$，$|y| < |[x|y]|$。因此正如所要求的那样，任何 $x, y, z, w \in H$，

$|a(y,z,w)| = \min\{|y|, |w|\} < |a([x|y],z,[x|w])| = \min\{|[x|y]|, |[x|w]|\}$

现在考虑(p2)的任何基本实例：

$$p(x,[y,z]) :- a(w,[y|v],x), a(w,v,u), p(u,z)$$

显然，

$$|p(x,[y,z])| = |x| + |[y|z]| + 1 > |a(w,[y|v],x)| = \min\{|w|, |x|\}$$

如果 $|w| + |[y|v]| \neq |x|$，那么根据我们所选取的 $\mathcal{M}$ 与可接受性的定义，定理得证。因而设 $|w| + |[y|v]| = |x|$，从而 $|w| \leqslant |x|$。因此 $|p(x, [y, z])| = |x| + |[y|z]| + 1 > |w| \geqslant |a(w, v, u)|$。定理再次得到证明，除非 $|w| + |v| = |u|$。在这一情形下，$|u| < |x|$ 且 $|p(x, [y, z])| = |x| + |[y|z]| + 1 > |u| + |[y|z]| + 1 = |p(u, z)|$，至此完成可接受性的证明。□

因为可接受性蕴涵左终止性，所以我们已经证明 PERM 采用 PROLOG 的标准运行在任何基本子句上都将会终止。根据习题 5，PERM 的逻辑后承正是该程序所期望的结果，所以 PERM 对于基本目标会终止并给出正确的答案。因此我们有一个被证实的方法可以检验一个列表是不是另一个的置换。更为有趣的是，我们能用有界性所表述的终止特征来证实这个方法可以做得更多。例如，由形如 $G = \{\neg p(x, X)\}$ 的目标开始，我们可以找出给定的列表 x 的所有置换。要看清这一点，只需证明 G 是有界的。我们证明一个更强的结果。

定理 4.13 对于 $\mathcal{L}$ 中的所有项 $t, t_1, \cdots, t_n$，每个目标 $G = \{\neg p([t_1, \cdots, t_n], t)\}$ 是有界的（对于定理 4.12 证明中的层映射和模型）。

证明 对于 G 的基本实例 $G\gamma$，$|G\gamma| = n + m + 1$，其中 $m = |t\gamma|$。因为对于任何基本替换 γ，$t\gamma$ 的长度是常数，所以 G 是有界的。□

可以证明，许多其他类型的目标子句对于 PERM 是有界的。这样的例子见习题 7。

习题

定义

(i) P **对于目标 G 是终止的**(terminating for a goal G)，如果 $P \cup \{G\}$ 的所有从 G 开始的 LD－证明是有穷的。P **是终止的**，如果它对于每个基本目标 G 是终止的。

(ii) P **对于层映射 f 是回归的**(recurrent with respect to a level mapping f)，如果对于 P' 中任何子句 $A:- A_1, \cdots, A_n$ 和每一个 $1 \leqslant i \leqslant n$，$|A| > |A_i|$。$P$ **是回归的**(recurrent)，如果 P 对于某个层映射是回归的。

1. 证明：如果 P 是回归的，那么它是终止的。
2. 证明：如果 P 是终止的，那么它是回归的。
3. 证明：PERM 不是终止的。

4. 证明：APPEND 是回归的。

5. 证明：对于任何 x，$y \in B$，PERM 的形如 $p(x, y)$ 的逻辑后承是该程序期望获得的结果。

6. 证明：我们可以用 p 和 a 在 B 上所期望的解释作为定理 4.12 的证明中所使用的模型。

7. 设 $G = \{\neg A_1, \cdots, \neg A_n\}$。证明：如果每个 A_i 都是(关于某个层映射和模型)有界的，那么 G 也是有界的。

第五节　相　　等

直到现在，我们完全忽略了数学中的相等关系。(注意在 PROLOG 中，“$t_1 = t_2$”用来表示 t_1 与 t_2 可以合一。)现在就让我们来看一看“真正”的相等，因为它确实是 PROLOG 的一个问题。语法上，我们引入一个特别的(专用的)二元谓词，用中缀法记为 $x = y$。(用“=”来表示相等是很普遍的，而在 PROLOG 语法中“=”有其他的用法，所以我们不得不说明一下。在除了 PROLOG 程序之外的所有上下文中，我们都用“=”表示相等关系。)现在必须扩展我们的推理方法以及谓词演算的语义以处理这个特殊的新谓词。

(在语言$\mathcal{L}$中)相等关系的基本性质可以表示为：

定义 5.1　$\mathcal{L}$的等词公理：

(1) $x = x$。

(2) $x_1 = y_1 \wedge \cdots \wedge x_n = y_n \rightarrow f(x_1, \cdots, x_n) = f(y_1, \cdots, y_n)$，其中 f 是$\mathcal{L}$的任何 n-元函数符号。

(3) $x_1 = y_1 \wedge \cdots \wedge x_n = y_n \rightarrow (P(x_1, \cdots, x_n) \rightarrow P(y_1, \cdots, y_n))$，其中 P 是$\mathcal{L}$的任何 n-元谓词符号(包括二元谓词“=”)。

(1)保证“=”具有自反性。相等关系的其他常见性质(对称性和传递性)由(3)可得，其中取 P 为“=”(习题 4)。注意这并不意味着“=”是真正的相等。

现在可以认为，我们用任何证明系统所处理的任何语句集合中都包含这些公理。因此，如果语句 S 的语言含“=”，那么 S 的一个表反驳是指 S^* 的一个表反驳，其中 S^* 是在 S 上添加关于 S 中所有函数和谓词符号的公理(1)~(3)的全称闭包而得的扩张。类似地，S 的一个消解反驳是指 S^* 的一个消解反驳。不幸的是，简单地加上这些子句会降低程序的效率。稍后考虑调解法等特殊化方法时，我们再回到这个问题。

下一步是要确定等号的指定语义。这里有两个选择。我们可以像对待其他谓词那样对待等号，仅仅把它解释为一个满足所有等词公理的二元关系。从证明论，即 PROLOG 的观点来看，这是我们有确实把握的唯一选择，并且在一个固定的语言范围之内，对于相等关系我们只能说出这么多。另一方面，从抽象的数学观点来看，我们希望一直把“=”解释为真正的相等关系。

事实上，我们可以将“=”解释为真正的相等，并且第二章第七节的基本定理仍然可以像以前一样证明：可靠性、完全性、紧致性等等。唯一的问题出现在完全性定理的证明中。在第二章定理 7.3 的证明中，由其中的 CST 构造了一个结构，它使得“=”满足所有等词公理，但是这并不能保证它就是真正的相等。解决的办法是用“=”所诱导的等价关系进行划分。具体来说，令$\mathcal{A}$为给定语句集 S 的 CST 的一条非矛盾路径所确定的结构。注意，$\mathcal{A}$的元素都是语言$\mathcal{L}$的基本项 t。定义这些元素上的等价关系“≡”为：$t_1 \equiv t_2 \Leftrightarrow \mathcal{A} \models t_1 = t_2$。应用等词公理易知，“≡”是等价关系(即对于任何 t，$t \equiv t$；如果 $t_1 \equiv t_2$，那么 $t_2 \equiv t_1$。并且如果 $t_1 \equiv t_2$ 且 $t_2 \equiv t_3$，那么 $t_1 \equiv t_3$)。然后用“≡”的等价类来定义$\mathcal{L}$的结构$\mathcal{B}$。即，$\mathcal{B}$的元素是所有形如

$[t_1]=\{t \mid t\equiv t_1\}$的集合，其中$t_1\in\mathcal{A}$。根据$\mathcal{A}$及选取等价类的代表元素，定义$\mathcal{B}$上的函数和关系如下$\mathcal{A}$：$\mathcal{B}\vDash P([t_1],\ \cdots,\ [t_n])\Leftrightarrow\mathcal{A}\vDash P(t_1,\ \cdots,\ t_n)$(其中$P$不为"="）且$f^{\mathcal{B}}([t_1],\ \cdots,\ [t_n])=[f^{\mathcal{A}}(t_1,\ \cdots,\ t_n)]$。显然，"="在$\mathcal{B}$中的解释是真正的相等。关于这一点，我们应当验证这些定义独立于代表元素的选取(即，从集合$[t_i]$中选取元素t_i)。最后，如同我们对$\mathcal{A}$所做的那样，用归纳法证明$\mathcal{B}$与用以构造$\mathcal{A}$的那条路径上的每一个带标记陈述一致。因此，$\mathcal{B}$是S的所要求模型且其中"="的解释是真正的相等。

定理 5.2(完全性) 如果S是一个语句集合且包含S的语言的等词公理，那么要么有从S出发的关于□的表证明，要么有S的一个模型，其中"="解释为真正的相等。

由于现在主要是从证明论，即消解方法的观点出发考虑问题，所以将构造的细节和对应于这些解释的紧致性定理证明留作习题。从现在起我们把"="仅仅视为一个专门的谓词符号。

定义 5.3 带"="**语言$\mathcal{L}$的一个等号结构**(equality structure for a language $\mathcal{L}$)是指$\mathcal{L}$的一个满足等词公理的结构。类似地，$\mathcal{L}$的**语句集S的一个等号模型**(equality model of a set of sentences S)指$\mathcal{L}$的一个等号结构，使得S的所有语句为真。从S出发的一个**等号消解(或者表)证明**(equality resolution (or tableau) proof)就是指来自S和等词公理的消解(或者表)证明。

根据定义，消解(或者表)证明系统的可靠性、完全性和紧致性定理对于等号的解释及相关证明也是成立的。然而考虑到实际执行带等号消解的效率，我们的处境要比原来糟糕得多。我们有了太多的新规则，对此我们给出一个修正的消解规则来处理等号。尽管如此，要缓解我们的问题，仍有相当长的一段路要走。

我们想要的推理方案(调解)将取代等号公理(2)和(3)。也就是说，我们想要一个(像消解那样的)规则，它与消解结合时，对于等号模型来说是完全的：如果$\{x=x\}\in S$且S没有等号模型，那么应用消解和调解由S可得□。(这里的要点是，S可以含有"="但除了$x=x$之外不包含其他的等词公理。)其基本思想是，如果有一个子句C_1含其中出现t的文字$L(t,\ \cdots)$和一个子句C_2(与C_1没有相同的变元)含$t=s$，那么从C_1和C_2所得到的不仅仅是$C_1\cup C_2$，还有在$C_1\cup C_2$中用s替换$L(t,\ \cdots)$中t的结果。当然我们不必在L中处处用s替换t。因此，我们考虑用s替换$L(t,\ \cdots)$中t的一次出现。(显然，通过反复使用这一规则可以实现多重替换。)我们用$L[t/s]$表示用s替换L中t的某个出现所得的结果。

例 5.4 从$C_1=\{\neg P(a),\ Q(b)\}$和$C_2=\{a=b,\ R(b)\}$得出$\{\neg P(b),\ Q(b),\ R(b)\}$。注意，结果中也扔掉了$a=b$。如同在消解中那样，它已被吸收到结果之中。

当然，像消解中那样，我们必须考虑合一运算所引入的可能性。

例 5.5 从$C_1=\{\neg P(x),\ Q(x)\}$和$C_2=\{b=b,\ R(b)\}$中，我们可以得出$\{\neg P(b),\ Q(b),R(b)\}$。这是用合一所作的简单替换。

再看一个更一般的例子：

例 5.6 从$C_1=\{\neg P(f(g(x))),\ Q(x)\}$和$C_2=\{g(h(c))=a,\ R(c)\}$中，可以得出$\{\neg P(f(a)),\ Q(h(c)),\ R(c)\}$。这里$g(x)$就是所考虑的项$t$。我们用替换$\{x/h(c)\}$将它与$g(h(c))$合一。在$C_1$上应用此替换得$\{\neg P(fgh(c)),\ Q(h(c))\}$后，根据$C_2$中的$g(h(c))=a$，再用$a$替换其中的$g(h(c))$，再结合$C_2$可得出我们的结果。

如同在消解那里一样，也可以在应用替换之前，用合一来合并文字。这里区分这两种操作是必要的。

定义 5.7 设我们可以重命名 C_1 和 C_2 中变元，使得两者没有相同的变元，并设 C_1 形如 $\{L(t_1), \cdots, L(t_n)\} \sqcup C'_1$ 且 C_2 形如 $\{r_1 = s_1, \cdots, r_m = s_m\} \sqcup C_2'$。如果 σ_1 是 $\{L(t_1), \cdots, L(t_n)\}$ 的 mgu，σ_2 是 $\{r_1 = s_1, \cdots, r_m = s_m\}$ 的 mgu 而 σ 是 $\{t_1\sigma_1, r_1\sigma_2\}$ 的 mgu，那么任何形如

$$\{L\sigma_1\sigma[t_1\sigma_1\sigma/s_1\sigma_2\sigma]\} \cup C'_1\sigma_1\sigma \cup C_2'\sigma_2\sigma$$

的子句是 C_1 与 C_2 的**调解式**(paramodulant)。

对于等号模型而言，这个规则与消解结合起来时是完全的。事实上，同消解一样，其线性版本也是完全的。

定理 5.8 如果 $\{x = x\} \in S$，$C \in S$ 且 $S - \{C\}$ 有一个等号模型但 S 没有，那么存在□的从 S 出发的由 C 开始的应用消解和调解规则的线性证明。(如你所期望的那样，这样的线性证明是一个序列 $\langle C_0, B_0\rangle, \cdots, \langle C_n, B_n\rangle$，其中 $C_0 = C$，$C_{n+1} = \square$，每一个 B_i 来自 S 或者是某个使得 $j < i$ 的 C_j，并且每一个 C_{i+1} 是由 C_i 和 B_i 应用一步消解或者调解得到的。)

该定理的证明与仅涉及消解的证明非常相似，我们在此略去。有关等号处理的一般问题是十分复杂的。它本身就是一个发展完备的研究领域，即规则重写。把等号与 PROLOG 或者更一般的定理证明器整合起来的问题还是当前研究中一个有待发展的课题，这已超出本书的范围。

习题

1. 证明：带等词公理的图表方法对于将"="视为真正的相等的解释是可靠的。
2. 对于将等号解释为真正相等的模型，给出(带等词公理的)表证明法完全性定理(定理 5.2)的一个完整的证明。
3. 对于将等号解释为真正相等的模型，证明：紧致性定理(一个语句集是可满足的，当且仅当它的每个有穷子集是可满足的)。
4. 由定义 5.1 中的(1)~(3)，证明："="的对称性和传递性。(提示：对于第一个，在(3)中令 $x_1 = x$，$x_2 = x$，$y_1 = y$ 且 $y_2 = x$。对于第二个，令 $x_1 = x$，$x_2 = y$，$y_1 = x$ 且 $z_2 = z$。)

第六节 因失败而否定

我们至今所描述的 PROLOG 还不能导出否定的信息。(使得 PROLOG 程序 P 的每个基本原子公式为真的厄布朗结构显然是 P 的一个模型。)然而，我们经常需要断言某个事实不成立。在 PROLOG 中，否定词往往用推理失败来执行，即目标$\neg A$(常写成"not(A)")成立，当且仅当 PROLOG 对目标 A 返回"no"。对于否定词的几种理解可以说明这种执行方式的合理性。最早的两个是封闭世界假设(closed world assumption，CWA)和完全数据库(completed database，CDB)观点。前者已超出了谓词逻辑的范畴，我们只作简要的解释；对于后者我们将作详细介绍。这两个方向也可以用于执行更一般的因失败而否定规则，它允许程序中的子句和目标子句可以是任意类型的子句。

下一节介绍否定词的一个比较新的研究方向，即稳定模型(stable models)。这些模型的特征也超出了谓词逻辑的范畴。它们与第七节中介绍的非单调逻辑(nonmonotonic logic)密切相关。

CWA 产生于对数据库的研究。如果有一份关于数学系学生成绩的数据库，我们一般会相信这是一个正确的且完全的成绩表。因此，如果表中没有出现这样的事实：琼斯的数学

486 的成绩是 A，那么可以合理地认为这个事实是假的。将这一原理推广到既有规则又有数据的 PROLOG 情形中，就有了程序 P 的 CWA 规则：如果某个基本原子公式(正文字)A 不是 P 的逻辑后承，那么可以推出 $\neg A$。

这里首先要注意的是，CWA 规则所处理的是抽象概念逻辑后承，或者等价地，在谓词逻辑的某个完全证明系统中的可证明性。然而，根据谓词逻辑可证明性的不可判定性(推论 8.10)，即使在理论上也不能执行这样的规则。我们所能期望的最接近的规则是，当有 A 不是 P 的逻辑后承的证明时可以得出结论 $\neg A$。对于 PROLOG 式的推理系统，这样的一个证明可以是一棵有穷的 SLD -树，它以 $\neg A$ 开头且每个分支都以失败结束。在这种情形中，不存在由给定的目标开始的 SLD -反驳。于是，由 SLD -反驳的完全性定理(定理 1.8)知，A 不是 P 的逻辑后承。这样的树称为 $P\cup\{A\}$ 的**有穷失败 SLD -树**(finitely failed SLD-tree)。

通常的 PROLOG 执行仅寻找一个带标准选择规则的有穷失败 SLD -树。然而为了便于理论分析，我们考虑一个更一般的、语义内涵更清楚的定义。为此，我们必须考虑这样的反驳搜索法，它不遵守标准选择规则甚至也不遵守总是从给定的目标子句中选择相同文字的选择规则。(见习题 2)。

我们从选择规则更一般的概念开始。这种规则使得，在证明的每一步中，文字的选取依赖于前面的整个证明过程，而不是仅仅依赖于当前的目标子句。

定义 6.1

(i) **一般化的选择规则**(generalized selection rule)R 是这样的一个函数，给定任何 LD -证明 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$，它从 G_n 中选取一个文字。

(ii) 一个 LD -证明 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$ 称为**带一般化选择规则 R 的证明**(proof via a generalized selection rule R)，如果在每一步 $i(0\leqslant i<n)$ 被消解的文字是通过 R 从当前的证明 $\langle G_0, C_0\rangle, \cdots, \langle G_i, C_i\rangle$ 中选取的。

对于给定的目标、程序和一般化选择规则，我们给出其对应 SLD -树的形式定义。

定义 6.2 令 P 为 PROLOG 程序，G 为某个目标子句而 R 为一般化的选择规则。**(从 P 出发由 G 开始带 R 的)关联 SLD -树**(associated SLD-tree(from P starting with G via R))是一个有穷分支树 T，其节点用目标子句标记，每条路径对应着一个带 R 的 SLD -证明。用归纳法定义 T。T 的根节点标记为 G。如果 T 的某个节点 σ 标记为目标子句 G'，并且一般化选择规则 R 根据从根节点到 G' 的那一段路径所对应的证明选取 G' 中的文字 $\neg A$，那么 σ 的各个直接后继分别对应着可以与 G' 在 $\neg A$ 上消解的 P 的子句 C_i。这些节点的标记分别是对应的消解式。(从根节点到达 G' 的某个后继节点的路径所对应的证明是，在从根节点到达 G' 的路径所对应的证明的后面，加上该后继所对应的消解。)

注意，一般地，P，G 和 R 对应的 SLD -树可以是无穷的。有些路径成功，即以□(成功)结束，从而是 G 的从 P 出发的(必然)有穷的反驳。有些路径失败，即 P 中没有子句可以与路径上最后面的目标在被选取的文字上消解。还可能有永不终止的其他路径。

为了逼近不是程序 P 的逻辑后承的文字所构成的集合，现在考虑这样的文字 A：寻找 A 的反驳的搜索在有穷步内失败。这样的失败是可以观察到的。

定义 6.3

(i) P，G 和 R 所对应的 SLD -树是**有穷失败的**(finitely failed)，如果它是有穷的且每条路径都是因为在 P 中找不到子句可以消解被选取的文字而终止。(特别地，没有路径以成功

即□结束。)

(ii)一个 PROLOG 程序 P 的 **SLD-有穷失败集**(SLD-finite failure set)是基本原子所构成的集合，满足对于其中每个基本原子 A，存在一般化选择规则 R，使得 P，$\{\neg A\}$ 和 R 所对应的 SLD-树是有穷失败的。

回忆 SLD-树的广度优先与深度优先两种搜索对系统完全性的不同影响。在这里，存在一般化选择规则 R 保证，如果 A 在 P 的有穷失败集中，那么其带 R 的 SLD-树是有穷失败的，但有些一般化选择规则是没有这个性质的。

定义 6.4 一个(带 R 的)SLD-证明是**公平的**(fair)，如果它要么是有穷的(因此是失败的或者是一个成功的反驳)，要么对于证明中出现的每一个文字 Q(比如说在 G_i 中)，R 在第 i 步选择 Q 或者在某个 $j>i$ 阶段选择 $Q\theta_i\cdots\theta_{j-1}$，其中 θ_k 是该证明第 k 步所使用的 mgu。一个一般化选择规则 R 是**公平的**，如果每个带 R 的 SLD-证明都是公平的。

设计一个公平的一般化选择规则并不困难。但是没有任何一个常规的选择规则是公平的。(见习题 1~2。)下面的结果说明我们只需关注任意公平的一般化选择规则。

定理 6.5 对于程序 P 和基本原子公式 A，A 在 P 的 SLD-有穷失败集中，当且仅当对于每个公平的一般化选择规则 R，A 的带 R 的 SLD-树是失败的。因此，存在 A 的一个使用任一选择规则的有穷失败 SLD-树，当且仅当对于任何公平的一般化选择规则 R，A 的 SLD-树是有穷失败的。

证明 充分条件立刻可从有穷失败集的定义得出。于是设 A 在 P 的有穷失败集中而 R 是任何公平的一般化选择规则。我们希望证明由 $\{\neg A\}$ 开始带 R 的 SLD-树是有穷失败的。我们用归纳法证明一个更强的引理。 □

引理 6.6 令 P 为 PROLOG 程序且 R 是公平的一般化选择规则。如果目标子句 $G=\{\neg A_1, \cdots, \neg A_m\}$ 有一个深度为 k(带任何一般化选择规则的)有穷失败 SLD-树且 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$ 是一个从 P 出发带 R 的 SLD-证明，其中 $G_n=G$，那么带 R 的 SLD-树的初始段为 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$ 的每一条路径都是有穷失败的。

证明 对 k 进行归纳。设 $k=1$ 且 A_s 是在当前深度为 1 的失败 SLD-树 S 中所选择的文字。根据失败 SLD-树的定义，P 中没有子句其头可以与 A_s 合一。现在考虑带 R 的 SLD-树 T 的初始段为 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$ 的路径 Q。由于 R 是公平的，R 在 Q 的某个节点上选取 $A_s\theta$，其中 θ 是一个替换。根据对 A_s 的假设，P 中没有子句其头可以与 $A_s\theta$ 合一。于是路径 Q 在该节点以失败结束。由于 Q 是 T 上具有那个特殊初始段的任意路径，因而 T 在此初始段以下是有穷失败的。

作为归纳步，令 A_s 为在当前深度为 $k+1$ 的有穷失败 SLD-树 S 的第一层中所选择的文字。于是，S 的第一层含有用 P 中子句消解 A_s 的所有结果。从而这一层上的每个节点的标记形如

$$H = \{\neg A_1, \cdots, \neg A_{s-1}, \neg B_1, \cdots, \neg B_k, \neg A_{s+1}, \cdots, \neg A_n\}\theta$$

其中 θ 是对应的消解和 P 中某子句 C 所使用的 mgu。注意 H 有一个深度为 k 的有穷失败 SLD-树。

现在令 Q 为带 R 的 SLD-树 T 上任意一条以 $\langle G_0, C_0\rangle, \cdots, \langle G_n, C_n\rangle$ 开头的路径，其中 $G_n=G$。再由 R 的公平性，在 Q 的例如第 $m-1$ 层中，首先选取形如 $A_s\psi$ 的文字，其中 $A_s\in G$。令 $\langle G_0, C_0\rangle, \cdots, \langle G_m, C_m\rangle$ 为 Q 上直到我们消解 $A_s\psi$ 为止的一段路径。此路径上

最后一个消解用到某个子句 $C\in P$，其头可以与 $A_s\psi$ 合一，从而也可以与 A_s 合一。交换引理(引理 1.11)的证明指出 G_m 是 S 的第一层中对应于 C_{m-1} 的那个子句 H 的改名。因为有穷失败 SLD－树不受改名的影响，所以有一个深度为 k 由 G_m 开始的有穷失败的 SLD－树。对 $\langle G_0, C_0\rangle, \cdots, \langle G_m, C_m\rangle$ 应用归纳假设可得 Q 是有穷失败的。 □

根据定理 6.5，在定义 SLD－树时可以假设已经指定了任意某个公平的一般化选择规则 R。现在可以弄清楚，PROLOG 类型的系统在配备公平的 R 之后，应当如何解决同时询问否定性结论和肯定性结论的问题。我们从同时含有正文字和负文字的子句 G 开始。然后我们进行带 R 的 SLD－证明，不同的地方在于当选取了正文字 A 时，我们就构造以 A 开头带 R 的有穷失败 SLD－树。如果成功，就删去 A 并继续。如果失败，则构造 G 的反驳的尝试也失败。我们将这个方法形式化为 SLDNF－反驳(带表示失败的否定词的 SLD－反驳)。

定义 6.7

(i) **一般目标子句**(general goal clause) G 仅是一个任意的子句。

(ii) 令 P 为 PROLOG 程序。一个**从 P 出发由 G 开始带 R 的 SLDNF－证明**(SLDNF-proof via R from P beginning with G)是一个序列 $\langle G_i, C_i\rangle$，其中 G_i 是一般目标子句，$C_i\in P$，$G_0=G$ 且 $G_{n+1}=\Box$，且该序列的生成规则如下：如果 $R(G_i)$(R 所选中的文字)是负的，那么 G_{i+1} 是 G_i 和 C_i 通过 mgu θ_i 在文字 $R(G_i)$ 上的消解式。如果 $R(G_i)$ 是正的，它一定是基本文字 A。在这种情况下，一定存在从目标 $\neg A$ 开始的(带 R 的)有穷失败 SLD－树。那么有 G_{i+1} 等于从 G_i 中删去 A，C_i 不起作用而 θ_i 是一个恒等映射。

像通常一样，这个证明所使用的 mgu 的复合序列 $\theta_0\cdots\theta_n=\theta$ 称为回答替换。

从 P 出发由 G 开始带 R 的 SLDNF－树的定义，除了(ii)中的改动之外，与对应的 SLD－树的定义相同(定义 6.2)。树上的一条路径就是构造这样 SLDNF－反驳的一个尝试。该路径成功，如果对应的尝试最终产生□。设在这条路径的某一点遇上一个以 $\neg R(G_i)$ 开头的 SLD－树 T，其中 $R(G_i)$ 是正的基本文字。如果 T 有一条路径以□结束，则 T 不是有穷失败的。这时我们说，这个寻找 SLDNF－反驳的尝试失败。(当然，即使该 SLDNF－树不是有穷失败的，我们可能永远不知道这个事实，且该证明方法可能陷入无穷搜索之中。)如果 $R(G_i)$ 是正的但不是基本的，我们说该证明在这个地方挣扎(flounder)。

提醒 仅当 R 选取的正文字是基本的，才允许 SLDNF－反驳进行下去。这样的选择称为安全的。这样的限制是根本性的，因为我们将 $\neg q$ 的成功解释为 q 的失败。如果 q 有一个自由变元 X，那么这样做是没有根据的。问题"?－$\neg q(X)$."要求一个 c，使得 $\neg q(c)$ 成立而"?－$q(X)$."要求一个 d，使得 $q(d)$ 成立。显然哪一个也不是另外一个的否定或者失败。遗憾的是，大多数 PROLOG 系统在应用表示失败的否定词方法时，不去检查一个正文字是不是基本的。这会导致意外的(实质上是失败的)结果。

在描述表示失败的否定词与 CWA 之间的关系之前，先介绍一个更一般的研究方向，称为"完全数据库"(CDB)。其思想是，当我们刻画了某件事发生的所有条件时，也就是完全刻画了这件事本身。下面用一个具体程序 P 来说明。设 r 是一个 n－元谓词且程序 P 的所有子句的头含 r：

$$r(t_{1,1},\cdots,t_{1,n}) :- q_{1,1},\cdots,q_{1,n_1}.$$

$$\vdots$$

$$r(t_{k,1},\cdots,t_{k,n}) :- q_{k,1},\cdots,q_{k,n_k}.$$

如果这个列表是对 $r(X_1, \cdots, X_n)$(X_i 为新变元)成立的完整描述，那么这一观点可以表示成

$$r(X_1,\cdots,X_n)\leftrightarrow q_1 \vee \cdots \vee q_k.$$

其中 X_1, …, X_n 为新变元，q_i 为 $\exists Y_1$, …, $Y_{p_i}(X_1 = t_{i,1} \wedge \cdots \wedge X_n = t_{i,n} \wedge q_{i,1} \wedge \cdots \wedge q_{i,n_i})$，这里 Y_1, …, Y_{p_i}分别是 $q_{i,1}$, …, q_{i,n_i}中的变元。这个等价式的“当”($\leftarrow$)方向仅是所给程序的各个断言。其“仅当”($\rightarrow$)方向说，我们已经用该程序的子句完全刻画了 r。对每个出现在 P 中的谓词 r，Comp(P)(P 的完备化，completion of P)包含上面的语句作为公理。如果 r 不出现在 P 任何子句的头，则公理 $\forall \vec{X} \neg r(\vec{X})$ 包含在 Comp(P)之中。在不涉及等号的情况下，Comp(P)只含有这些公理。

在涉及等号的情况下，Comp(P)包含第五节中所有在 P 的语言中的基本等词公理(1)~(3)。此外，数据库观点指出不同的术语(名字)表示不同的事物。为了(在一阶逻辑允许的范围内)体现这一观点，令 Comp(P)也包含下面的公理：

(4)$f(x_1, \cdots, x_n) \neq g(y_1, \cdots, y_m)$，对于每一对不同的分别为 n 元和 m 元的函数符号 f 和 g。

(5)$t(x) \neq x$，对于每个含 x 但不同于 x 的项 $t(x)$。

(6)$f(x_1, \cdots, x_n) = f(y_1, \cdots, y_n) \rightarrow x_1 = y_1 \wedge \cdots \wedge x_n = y_n$，对于每个 n-元函数符号 f。

这就完成了从 P 到 Comp(P)的构造。P 的每个子句显然是 Comp(P)的逻辑后承，从而有 Comp(P)$\models P$。另外，如果 P 是一个 PROLOG 程序，Comp(P)是相容的。(这里，那些将“=”解释为真正相等并且使得其他基本原子公式都为真的厄布朗结构再一次成为模型。)现在用 Comp(P)证明有关定理的可靠性与完全性，以“表明”因失败而否定规则的合理性。我们从论述合一与等词公理(1)~(6)关系的引理开始。

引理 6.8 令 $S = \{s_1 = t_1, \cdots, s_n = t_n\}$。

(i)如果 S 是可合一的且 $\theta = \{x_1/r_1, \cdots, x_m/r_m\}$ 为合一算法(第二章算法 12.3)所给出的 mgu，那么(1)~(6)$\models (s_1 = t_1 \wedge \cdots \wedge s_n = t_n) \rightarrow (x_1 = r_1 \wedge \cdots \wedge x_m = r_m)$。

(ii)如果 S 不可合一，那么对任何语句 A，(1)~(6)$\models (s_1 = t_1 \wedge \cdots \wedge s_n = t_n) \rightarrow A \wedge \neg A$。

证明 考虑应用于 S 的合一算法。它产生替换序列 θ_0, θ_1, …, θ_n。令$\{x_1/r_{k,1}, \cdots, x_m/r_{k,m}\}$为替换序列的复合 $\theta_0 \cdots \theta_k$。对 k 进行归纳证明，对于直到合一算法终止时的每一个 k，(1)~(6)$\models (s_1 = t_1 \wedge \cdots \wedge s_n = t_n)\theta_0 \cdots \theta_k \rightarrow (x_1 = r_{k,1} \wedge \cdots \wedge x_m = r_{k,m})$。如果算法终止时给出一个合一替换，那么就证明了(i)。如果算法终止时宣告 S 不可合一，易知(1)~(6)可证明$(s_1 = t_1 \wedge \cdots \wedge s_n = t_n)\theta_0 \cdots \theta_k$ 没有实例为真，这样就证明了(ii)。我们将归纳和验证的细节留作习题 3。 □

定理 6.9(SLDNF-反驳的可靠性) 令 P 为 PROLOG 程序。

(i)如果从 P 出发由形如$\{L_1, \cdots, L_m\}$的一般目标 G 开始的带 R 的 SLDNF-树是有穷失败的，那么 Comp(P)$\models L_1 \vee \cdots \vee L_m$。

(ii)如果该树上存在一条带回答替换 θ 的成功路径，即 G 的从 P 出发的 SLDNF-反驳，那么 Comp(P)$\models (\neg L_1 \wedge \cdots \wedge \neg L_m)\theta$。

证明 (i)对由 G 开始的有穷失败 SLDNF-树的深度进行归纳证明。先考虑该失败树在第一层上是有穷失败的。

如果 $R(G)$是一个负文字 $L = \neg r(s_1, \cdots, s_n)$，则没有子句 $C \in P$，满足其头可以与

$r(s_1, \cdots, s_n)$合一。如果 P 中没有子句其头含 r，则 Comp(P)包含公理 $\forall\vec{X}\neg r(\vec{X})$，从而 Comp(P)$\models\neg r(s_1, \cdots, s_n)$。否则，应用定义中的符号表示，有 Comp($P$)$\models r(s_1, \cdots, s_n)\leftrightarrow\vee\{\exists Y_1\cdots\exists Y_{p_i}(s_1=t_{i,1}\wedge\cdots\wedge s_n=t_{i,n}\wedge q_{i,1}\wedge\cdots\wedge q_{i,n_i})\mid i\leqslant k\}$。因为根据假设，$r(s_1, \cdots, s_n)$不能与任何 $r(t_{i,1}, \cdots, t_{i,n})$合一，所以根据引理 6.8(ii)由等词公理可得，该式中每一个析取项都不成立。因此 Comp(P)$\models\neg r(s_1, \cdots, s_n)$，从而 Comp($P$)$\models G$。

如果 $R(G)$是一个正文字 L，那么它一定是基本的(否则证明将会挣扎而不是失败)，且从$\neg L$ 开始的 SLD-树一定有一条路径以□结束。因此根据 SLD-反驳的可靠性(定理 1.9)可得所要求的结果 $P\models L$。(注意，由于 L 是基本的，该反驳所给出的回答替换是无关紧要的。)

现在考虑归纳步。设 G 有一个深度为 $k+1$ 的有穷失败 SLD-树。如果 $R(G)$是一个正的基本文字 L，那么从$\neg L$ 开始的 SLD-树是有穷失败的且 G_1 为 $G-\{L\}$。它有一个深度为 k 的有穷失败 SLDNF-树，从而根据归纳，Comp(P)$\models G_1$。因为 G 包含 G_1，Comp(P)$\models G$ 也成立。

最后，设 $R(G)$是负文字 $L=\neg r(s_1, \cdots, s_n)$。(再一次采用 Comp($P$)定义中的符号表示。)在所给的失败 SLDNF-树的第一层上，G 的每一个直接后继 H_i 是应用相应的 mgu θ_i 于 G，将 L 替换为$\neg q_{i,1}, \cdots, \neg q_{i,n_i}$(对 $i\leqslant k$)的结果。每一个 H_i 都有深度$\leqslant k$ 的有穷失败 SLDNF-树，从而根据归纳，对于每一个 $i\leqslant k$ 有 Comp(P)$\models H_i$。因此只需证明 Comp(P)$\models\wedge H_i\rightarrow\forall\vec{X}G$。要看清这一点，又只需证明

$$\mathrm{Comp}(P)\models\wedge\{(\neg q_{i,1}\vee\cdots\vee\neg q_{i,n_i})\theta_i\mid i\leqslant k\}\rightarrow\neg r(s_1,\cdots,s_n)$$

现在根据 Comp(P)的定义，$\neg r(s_1, \cdots, s_n)$不成立，仅当对某个 $i\leqslant k$，有 $\exists Y_1\cdots\exists Y_{p_i}(s_1=t_{i,1}\wedge\cdots\wedge s_n=t_{i,n}\wedge q_{i,1}\wedge\cdots\wedge q_{i,n_i})$。根据引理 6.8，这一命题成立的必要条件是，存在$\vec{Y}$，对每个$j\leqslant n$合一 s_j 与 $t_{i,j}$，并且保证 $q_{i,1}\wedge\cdots\wedge q_{i,n_i}$成立。由于 θ_i 是这个合一的 mgu，所以由假设$(\neg q_{i,1}\vee\cdots\vee\neg q_{i,n_i})\theta_i$ 可知不存在这样的$\vec{Y}$，得证。 □

(ii)对有穷失败 SLDNF-反驳的长度进行归纳证明。首先，设反驳长度为 1，从而 G 只含一个文字 L。如果 L 是正的，它就是基本的且有一个从$\neg L$ 开始的有穷失败 SLD-树。根据(i)得 Comp(P)$\models\neg L$。如果 L 是负的，例如$\neg r(s_1, \cdots, s_n)$，那么 P 中有一个形如$r(t_1, \cdots, t_n)$的子句可以通过某个 θ 与 L 合一。因此，$\neg L\theta$ 是 P 的从而也是 Comp(P)的逻辑后承。

然后，考虑 G 的长度为 $k+1$ 的 SLDNF-反驳。如果 $R(G)$是一个正文字 L_i，那么 L_i 是基本的，θ_0 是恒等映射，G_1 形如$\{L_1, \cdots, L_{i-1}, L_{i+1}, \cdots, L_m\}$且有一个长度为 k 的 SLDNF-反驳，其 mgu 序列记为 $\theta_1\cdots\theta_k$。与基本情形一样，Comp(P)$\models\neg L_i$，根据归纳假设有 Comp(P)$\models(\neg L_1\wedge\cdots\wedge\neg L_{i-1}\wedge\neg L_{i+1}\wedge\cdots\wedge\neg L_m)\theta_1\cdots\theta_k$。由此可知所要证明的 Comp($P$)$\models(\neg L_1\wedge\cdots\wedge\neg L_m)\theta_0\theta_1\cdots\theta_k$ 成立。

最后，设 $R(G)$为一负文字 $L_i=\neg r(s_1, \cdots, s_n)$，且 $G_1=\{L_1, \cdots, L_{i-1}, \neg q_{j,1}, \cdots, \neg q_{j,n_j}, L_{i+1}, \cdots, L_m\}\theta_0$。根据归纳，Comp($P$)$\models\{\neg L_1\wedge\cdots\wedge\neg L_{i-1}\wedge q_{j,1}\wedge\cdots\wedge q_{j,n_j}\wedge\neg L_{i+1}\wedge\cdots\wedge\neg L_m\}\theta_0\cdots\theta_{k+1}$。现在根据 Comp($P$)的定义、$\theta_0$ 合一 $r(s_1, \cdots, s_n)$与 $r(t_{j,1}, \cdots, t_{j,n})$的事实和引理 6.8(i)，有 Comp($P$)$\models(q_{j,1}\wedge\cdots\wedge q_{j,n_j})\theta_0\rightarrow r(s_1, \cdots, s_n)\theta_0$。因此可知 Comp($P$)$\models\neg L_i\theta_0\cdots\theta_k$成立，从而结束归纳步。 □

定理 6.10(SLDNF-反驳的完全性) *如果 P 是一个 PROLOG 程序，$G=\{\neg A_1, \cdots, \neg A_k\}$是常规的目标子句，$R$ 是公平的一般化选择规则且 Comp(P)$\models\neg A_1\vee\cdots\vee\neg A_k$，那么存在一*

个从 P 出发由 G 开始的带 R 的有穷失败 SLD-树。

证明 为了处理第五节中的等号，下面的证明思路修正了第二章第七节的表方法完全性定理的证明。如果从 G 开始的 SLD-树不是有穷失败的，那么它有一条(可能是无穷的)路径 Q。用 Q 上出现的项和公式定义 Comp(P)的模型$\mathcal{M}$，使得$\mathcal{M} \vDash \exists \vec{X}(A_1 \wedge \cdots \wedge A_k)$。($\vec{X}$列出了 G 的所有变元。)因此，通过构造这样的模型$\mathcal{M}$，可得所要证明的定理的逆否命题。

令 G_0，G_1，…和 θ_0，θ_1，…分别为出现在 Q 所对应的 SLD-证明中的目标和 mgu。如同第五节中那样，M 的元素是项的等价类，并且函数符号在这些等价类上的作用也同那里所定义的一样。这里关键的新思想是用 mgu 序列定义项的等价关系。我们说两个项 s 和 t 是等价的($s \equiv t$)，如果它们最终会被 mgu 序列所合一，即存在 m，使得 $s\theta_0 \cdots \theta_m = t\theta_0 \cdots \theta_m$。显然这一关系是等价关系(习题4)。我们把项 t 的等价类记为$[t]$，并令所要定义的模型的全域 M 为这些等价类集合。显然，当“=”解释为真正相等时，Comp(P)的等词公理(1)~(6)在 M 中成立(习题5)。

为了定义哪些原子公式在$\mathcal{M}$中为真，第一步规定：$r([t_1], \cdots, [t_n])$为真，如果存在 $s_i \in [t_i]$，使得$\neg r(s_1, \cdots, s_n)$是某个目标子句 G_m 中出现的文字。注意，由此立即可得$\mathcal{M} \vDash \exists \vec{X}(A_1 \wedge \cdots \wedge A_k)$(因为 G 中项(的等价类)提供了证据)。

我们下一个也是最重要的断言是，至今所定义的原子事实集 S 满足 Comp(P)中关于谓词字母公理的“仅当”方向。设$\neg r(s_1, \cdots, s_n)$是首先出现在目标子句 G_m 中的文字。由 R 的公平性，存在 $u > m$，使得第 u 个被选择的文字为$\neg r(s_1, \cdots, s_n)\theta_m \cdots \theta_u$。注意根据选取变元和 mgu 的惯例，$\neg r(s_1, \cdots, s_n)\theta_m \cdots \theta_u = \neg r(s_1, \cdots, s_n)\theta_0 \cdots \theta_u$。在这一步，$G_m$ 中所选取的文字被 P 中某个子句体的文字$(\neg q_{i,1}, \cdots, \neg q_{i,n_i})\theta_{u+1}(=(\neg q_{i,1}, \cdots, \neg q_{i,n_i})\theta_0 \cdots \theta_{u+1})$所替换。由于 θ_{u+1}是这一消解的 mgu，从而每个 $q_{i,j}\theta_{u+1}$都在 S 中。所以，如果令 q_i 为 Comp(P)中对应于$r([s_1], \cdots, [s_n])$的公理实例的析取项，则根据引理 6.8(i)，有 q_i 是可满足的证据。

现在扩展 S，使得在保证$\mathcal{M}$满足公理“仅当”方向的同时还是公理“当”方向的模型，也就是程序 P 的模型。令 P'为用 M 的元素对 P 中子句做基本替换所得的实例集，且 S'为 $S \cup P'$的消解式集。令$\mathcal{M}$使得 S'是在 M 中为真的原子事实集。我们断言$\mathcal{M}$就是所要求的 Comp(P)的模型。由于它显然是 P 的模型，则只需验证每个谓词 r 所对应的公理的“仅当”方向在$\mathcal{M}$中也为真。通过对将 $r(t_1, \cdots, t_n)$放入 S'的消解推理的长度进行归纳证明，立刻可以得出这个结论。因为将$r(t_1, \cdots, t_n)$放入 S'就意味着，在 P 有某个头含 r 的子句，该子句中 $q_{i,1}$，…，q_{i,n_i}的一些适当的实例已经被推导出来。 □

Comp(P)的定义形式化了 CDB 研究方法背后的直觉。类似地，CWA(P)是由根据 CWA(封闭世界假设)应当与 P 关联的那些语句所构成的集合。CWA 的基本直觉是，对任何正的基本文字 L，如果 $P \nvdash L$，那么我们应得出$\neg L$。因此我们可以将此视为对 P 附加了下面类型的子句：

(0) $\{\neg L\}$，对每个基本正文字 L，使得 $P \nvdash L$。

和 CDB 一样 CWA 认为基本项足以正确地表达论域，但 CWA 更进一步。除了上面给出的等词公理(1)~(6)之外，它还声称全域恰好由这些基本项构成。然而，这一观点不能用谓词逻辑公式表示(习题6)。如果考虑逻辑后承($\vDash$)但不考虑可证明性，那么可以将这个要求表

示为一个无穷子句 DCA，论域闭包公理(domain closure axiom)：

(7) $$x=t_1 \vee \cdots \vee x=t_n \vee \cdots$$

其中$\langle t_i \rangle$是所有基本项的列表。

现在用 CWA(P)表示根据(0)～(7)对 P 所作的扩张。注意到 CWA(P)的任何模型都是 P 的厄布朗模型。由于附加上(0)可保证每个基本文字的真值由 CWA(P)决定，所以 CWA(P)至多有一个这样的模型。事实上，对于任何 PROLOG 程序 P，CWA(P)总是可满足的且它的唯一模型是 P 的最小厄布朗模型(习题7)。由于这个模型也是 Comp(P)的模型(习题8)，针对 Comp(P)和因失败而否定规则所证明的可靠性结果(定理6.9)自然对 CWA(P)成立。然而，可能没有与定理6.10对应的关于 CWA(P)的完全性定理。事实上，没有能行的方法(例如搜索 SLDNF-反驳)可以对每个 P 列出 CWA(P)的所有逻辑后承(第八节习题6)。

除了有时要得出否定信息，PROLOG 程序员也可能在程序中使用这样的表达式。这需要给出一般程序和一般 SLDNF-消解的概念。

定义 6.11 **一般程序子句**(general program clause)至少含有一个正文字(可以有更多)。**一般程序**(general program)是一般程序子句的集合。

在给定的一般程序子句$\{R, \bar{L}_1, \cdots, \bar{L}_n\}$中，确定其中一个正文字 R 为头而其他的文字作为子句的体。这样我们可以用 PROLOG 的(带¬的)表示方法把该子句写成 $R:-L_1, \cdots, L_n$。(遗憾的是，其解释和分析要依赖于哪一个正文字被选作子句头。)同样，我们把一般目标子句写成$\{\neg L_1, \cdots, \neg L_n\}$。然而同以前一样，这里希望用一些 mgu 的 $\theta_0 \cdots \theta_k=\theta$(通过某种形式的消解)从$P \cup \{G\}$得到□，从而证明 $P \vDash \exists X_1 \cdots X_m(L_1 \wedge \cdots \wedge L_n)\theta$。

通过在寻找有穷失败树的地方引入递归(recursion)概念，我们可以把 SLDNF-反驳的定义扩展到一般程序上。现在寻找一个有穷失败 SLDNF-树。把一般程序 P 扩展成 CWA(P)和 Comp(P)的方法与以前的定义相同。像定理6.8和定理6.9那样的可靠性结果也可以在这个一般背景下得到证明。但是定理6.10的完全性结果不再成立。事实上，甚至对于所有 PROLOG 程序，完全性定理也是不能推广到一般目标子句的(习题9)。只考虑各分支终止于成功或者失败的 SLDNF-树，可以得出比较弱的完全性结果。在这些条件下，有可能证明 SLDNF-树，在一定的意义下给出了"所有的"答案，它们是 CWA(P)或 Comp(P)的结果。我们建议读者参阅 Shepherdson[1992, 5.4]中关于 CWA(P)的部分和 Lloyd[1987, 5.4]第三章中关于 CDB 和 Comp(P)的详细讨论。

对于一般程序 P，需要特别注意的是，即使 P 是可满足的，有可能 CWA(P)或 Comp(P)或两者一起是不可满足的(习题10～12)。然而，第四节所考虑的回归性与可接受性的条件可以用以保证 Comp(P)的相容性，从而可得对于 Comp(P)所确定的语义而言的 SLDNF-反驳完全性定理。(这里我们再次建议读者参阅 Lloyd[1987, 5.4]。)

习题

1. 证明：总是从每个目标子句中选择同一个文字的选择规则不是公平的。
(提示：考虑由以下三个子句构成的程序 P：

$$(1)\ r:-p,q. \qquad (2)\ p:-p. \qquad (3)\ q:-q.\ ).$$

2. 给出一个公平的一般化选择规则并证明它是公平的。(提示：总选择证明中还没被选到的最先出现的文字。)

3. 完成引理 6.8 的证明。
4. 验证定理 6.10 的证明中所定义的关系“≡”是等价关系。
5. 验证当“=”解释为等价类之间的真正相等关系时，等词公理(1)~(6)在定理 6.10 的证明中所定义的集合 M 中被满足。
6. 证明：谓词逻辑的任何语句集都不能推出 CWA 的公理(7)。(提示：使用紧致性定理。)
7. 对于 PROLOG 程序 P，证明：CWA(P)的唯一的模型是 P 的最小厄布朗模型。
8. 证明：一个 PROLOG 程序的最小厄布朗模型也是 Comp(P)的模型。
9. 给出定理 6.10 向一般目标子句推广的反例。(提示：写一个短小的程序并选择某个一般目标子句，使得每个 SLDNF-反驳的尝试都将处于挣扎状态。)
10. 给出一个一般程序 P，使得 Comp(P)(从而 P)可满足但 CWA(P)不可满足。
11. 给出一个一般程序 P，使得 CWA(P)(从而 P)可满足但 Comp(P)不可满足。
12. 给出一个可满足的一般程序 P，使得 Comp(P)和 CWA(P)都不可满足。

第七节　否定和非单调逻辑

上一节描述的因失败而否定规则的一般执行方法既有用又重要，然而它违背数学推理的最基本原则之一。在数学推理(以及本书的所有其他系统)中从一个前件集合推出的结论也可以从任何更大的前件集合推出。更多的信息或者公理不会使已经做过的推理无效。推理的这种单调性对于标准的数学推理而言是基本的，然而现实生活的许多推理方法包括因失败而否定规则却没有这个性质。如果没有反面的证据，我们通常会接受某个与我们原来信念不矛盾的新信念。举一个小鸟 Tweety 的例子。在我们知识发展的某个阶段，通过观察认识了不同的鸟类。在这些信息的基础上，我们得出结论：鸟儿是会飞的。一天当我们听说某个名叫 Tweety 的小鸟时，自然想到它是会飞的。后来当我们看到 Tweety 时，才发现它原来是一只宠物鸵鸟，同我们一样不会飞翔。我们就抛弃了原先鸟儿会飞的信念，并更正我们对于 Tweety 的看法。现在带着新的信念面对这个世界，继续我们的推理，直到再次发现推翻我们信念的新证据。这是除了数学以外的几乎所有学科典型的知识增长过程。信念与结论常常以缺少其对立面的证据为基础。

因失败而否定的观念体现了类似的想法。如果没有反面的证据(即得出 L 的推理)，则假设 L 是不成立的。这个推理方法显然就是一个非单调的推理系统。Minsky[1975，5.5]首先提出这样的系统，而从麦卡锡(McCarthy)的限定论(study of circumscription)[1980，5.5]开始，许多研究人员针对计算机科学和人工智能中的不同问题提出和研究了大量的非单调系统。这里仅列出其中几个：Hintikka 的多重信念者理论(theory of multiple believers)，Doyle 的真理维护系统(truth maintenance system)，Reiter 的缺省逻辑(default logic)和 Moore 的自认知逻辑(autoepistemic logic)，还有 Apt，Clark 等人在扩展 PROLOG 中所提出的各种因失败而否定规则。

我们现在简要地说明由 Marek，Nerode 和 Remmel[1990，5.5]给出的、从抽象观点研究非单调逻辑系统的新方法。这一方法似乎抓住了所提到的许多系统的共同内涵。该文献主要处理命题逻辑的情形，我们也仅限于这个情形。对于 PROLOG 中的否定来说，它意味着，我们总是考虑给定程序在某个厄布朗域中的基本实例集。在描述该一般系统后，我们把它与一个有趣的方法联系起来，该方法可以挑选出一个特别的厄布朗模型用以体现 PROLOG 中否定的许多性质(尽管该方法与第六节中因失败而否定规则不完全相同)：Gelfond 和 Lifschitz

的稳定模型语义(stable model semantics)[1988, 5.4]。

我们把非单调系统的思想用推理规则的形式表示出来，就像消解或者第一章第七节中经典单调逻辑的推理规则那样。一个推理规则由一列假设和可从这些假设得出的一个结论构成。标准的假言推理规则(第一章规则 7.2)从假设 α 和 $\alpha \to \beta$ 得出 β。描述这个规则的一种恰当方式是：把假设写在一条直线的上面，把结论写在下面：

$$\frac{\alpha, \alpha \to \beta}{\beta}$$

在这样的表示中，公理就是不带假设的规则，例如第一章公理 7.1(i)中的公理：

$$\overline{(\alpha \to (\beta \to \alpha))}$$

将这样的系统推广到非单调逻辑的关键是给推导添加约束条件。为了得出规则指定的结论，除了要知道假设集合中的每个命题之外，也许还需要不知道(不相信、还没有证明、还没有得到等等)另外某个命题集合中的每个命题。这种规则的表示把常规的前件写在前面，把约束条件写在后面，中间用冒号隔开。约束条件是该规则要求我们不知道(不相信等等)的一些命题。因此我们把下面的规则

$$\frac{\alpha_1, \cdots, \alpha_n : \beta_1, \cdots, \beta_m}{\gamma}$$

读作：如果已经知道(证明、得到)α_1，…，α_n，但不知道(没证明、没得到)β_1，…，β_m，那么可以得出结论说我们知道(能证明或者得到)γ。

定义 7.1(非单调形式系统) 令 U 为(命题字母)集。

(i)**非单调推理规则**(nonmonotonic rule of inference)是一个三元组 $\langle P, G, \varphi \rangle$，其中 $P = \{\alpha_1, \cdots, \alpha_n\}$ 和 $G = \{\beta_1, \cdots, \beta_m\}$ 为 U 中元素的有穷列，且 $\varphi \in U$。每个非单调推理规则记为

$$r = \frac{\alpha_1, \cdots, \alpha_n : \beta_1, \cdots, \beta_m}{\varphi}$$

我们称 α_1，…，α_n 为规则 r 的**前件**(premise)，β_1，…，β_m 为规则 r 的**约束**(restraint)。注意 P 或 G 或两者同时可以为空。

(ii)如果 $P = G = \varnothing$，那么规则 r 称为**公理**。

(iii)**非单调形式系统**(nonmonotonic formal system)是一个有序对 $\langle U, N \rangle$，其中 U 为非空(命题字母)集且 N 为非单调规则集。

(iv)U 的子集 S 在系统 $\langle U, N \rangle$ 中是**推理封闭的**(deductively closed)，如果对于 N 的每个规则 r，若 r 的所有前件 α_1，…，α_n 都在 S 中且所有约束 β_1，…，β_m 都不在 S 中，则 r 的结论 φ 在 S 中。

形式系统的非单调性的实质在于推理封闭集对于任意交一般是不封闭的。因此一般来说，一个命题字母集 I 没有推理闭包，即没有最小推理封闭集 S，使得 $S \supseteq I$。然而，推理封闭集的递减序列的交还是推理封闭的(习题 1)，从而 U 的子集中总有(至少一个)极小推理封闭集(习题 2)。

在假设 I 的所有元素为真时，包含 I 的所有推理封闭集可看作是给定系统所允许的各种合理观点。每一个观点表达了一个在所有规则下封闭的信念集。然而，许多观点可能会相互矛盾。所有包含 I 的推理封闭集的交集代表这些合理观点的共同信息。它常被称为 I 的确定

后承(secured consequence)集或者系统与 I 的可疑推理(skeptical reasoning)结果集。

例 7.2 令 $U=\{\alpha, \beta, \gamma\}$ 且令

$$r_1 = \frac{}{\alpha} \quad r_3 = \frac{\alpha : \beta}{\gamma}$$

$$r_2 = \frac{\alpha : \beta}{\beta} \quad r_4 = \frac{\alpha : \gamma}{\beta}$$

(i)令 $N_1=\{r_1, r_2\}$。对于 $\langle U, N_1\rangle$：$S=\{\alpha, \beta\}$，只有一个极小推理封闭集。那么 S 是 $\langle U, N_1\rangle$ 的确定后承集。

(ii)令 $N_2=\{r_1, r_3, r_4\}$。对于 $\langle U, N_2\rangle$：$S_1=\{\alpha, \beta\}$ 和 $S_2=\{\alpha, \gamma\}$，有两个极小推理封闭集。那么 $S=\{\alpha\}$ 是 $\langle U, N_2\rangle$ 的确定后承集。在这个例子中，确定后承集不再是推理封闭的。

对应于经典的概念，非单调逻辑中从前件集 I 的推导需要一个参数 S，用来界定那些我们假设不知道的命题。我们用这种演绎的概念来刻画非单调系统的各个扩张(extension)，它类似于单调系统的逻辑后承集。

定义 7.3 *令 $\langle U, N\rangle$ 为非单调形式系统且令 S，$I\subseteq U$。*在 $\langle U, N\rangle$ 中 φ 的一个从 I 出发的 S-演绎*是一个有穷序列 φ_1，…，φ_k，使得 $\varphi=\varphi_k$ 且对所有 $i\leqslant k$，φ_i 要么在 I 中，要么是 $\langle U, N\rangle$ 的一个公理，要么是某个规则 $r\in N$ 的结论，满足该规则的所有前件在 φ_1，…，φ_{i-1} 中且所有约束在 $U-S$ 中。在这种情况下 φ 称为* **I 的 S-后承**，*我们用 $C_S(I)$ 标记 I 的所有 S-后承构成的集合。*

注意上述定义中 S 的作用是阻止应用其约束在 S 中的规则；它不会导致 U 的成员被直接放入 $C_S(I)$。事实上，$C_S(I)$ 有可能不包含 S 并且不是推理封闭的。

例 7.4 使用例 7.2 中的符号表示，定义一个系统 $\langle U, N\rangle$，其中 $N=\{r_1, r_3\}$。如果 $S=\{\beta\}$，那么 $C_S(\varnothing)=\{\alpha\}$ 不是推理封闭的，因为它不含 γ，而这与规则 r_3 抵触。

命题 7.5 *如果 $S\subseteq C_S(I)$，那么 $C_S(I)$ 是推理封闭的。*

证明 设规则 r 的结论为 φ，所有前件在 $C_S(I)$ 中而所有约束在其外面。根据 $C_S(I)$ 的定义，我们可以构造含 r 所有前件的 S-演绎。根据假设，r 的所有约束都在 S 的外面。因此应用 r 可以将该推导扩张为 φ 的 S-演绎，从而得 $\varphi\in C_S(I)$。 □

定义 7.6 *$S\subseteq U$ 是* **I 的一个扩张**，*如果 $C_S(I)=S$。S 是一个扩张，如果它是空集 $\varnothing$ 的扩张。*

非单调系统中 I 的扩张 S 对应于经典逻辑中 I 的逻辑后承。从 I 可推导出其扩张中的每一个成员，而 I 的所有 S-后承都在 S 中。在习题 3 ~ 5 中给出扩张的一些基本性质。

已经知道，可以应用扩张这个概念解决数学和计算机科学中的许多问题。在习题 8 ~ 9 中给出一些数学例子。现在回到带否定的 PROLOG 程序，通过稳定模型概念探讨它们与扩张的联系。

从我们现在的观点来看，自然要把因失败而否定视为一种非单调系统。上一节给出的因失败而否定的内在思想是当我们不知道(不能推出)p 时，可以断言 $\neg p$。这给出了一种将 PROLOG 程序转变成非单调系统的自然方式。

注意，定义 6.11 中一般程序子句的形式为 $p{:}-q_1$，…，q_n，$\neg s_1$，…，$\neg s_m$，其中 p，q_i 和 s_j 为原子公式。

不要忘记我们是在命题情形中。因此，如果所考虑的程序有变元，那么就考虑该程序子句在厄布朗域中的所有基本实例。我们现在可以轻易地把只含基本原子的一般程序 P 以一种自然方式翻译为非单调系统。我们把每个基本原子视为命题字母。这些原子构成了全域 U。把程序 P 的每个形如 $p:-q_1,\cdots,q_n,\neg s_1,\cdots,\neg s_m$ 的程序子句 C 翻译成一个规则 $\mathrm{tr}(C)$：

$$\frac{q_1,\cdots,q_n:s_1,\cdots,s_m}{p}$$

这样就可以得到一个非单调系统，其规则集 N 是翻译 P 的子句所得的集合 $\{\mathrm{tr}(C):C\in P\}$。

定义 7.7 令 P 为只含基本子句的一般程序。P 的**翻译**(translation) $\mathrm{tr}(P)$ 是非单调系统 $\langle U,N\rangle$，其中 U 是 P 中出现的原子所构成的集合，而 $N=\{\mathrm{tr}(C):C\in P\}$ 是 P 中子句的翻译所构成的集合。

已经证明 $\mathrm{tr}(P)$ 的扩张恰好就是 Gelfond 和 Lifschitz[1988, 5.4]所引入的 P 的稳定模型。稳定模型给带因失败而否定规则的一般程序提供了一种强语义概念。

定义 7.8 如果 U 是在一般的基本程序 P 中出现的原子所构成的集合且 $M\subseteq U$，那么 P_M 是用如下的方式由 P 得出的程序：对于 $s\in M$ 删除所有体部含有负文字 $\neg s$ 的子句，并且删去其他子句体的所有负文字。因为 P_M 显然是一个 PROLOG 程序(不含负文字)，根据第二章第十节习题 3，它有唯一的极小厄布朗模型。P 的**稳定模型**(stable model)是 U 的一个子集 M，使得 M 是 P_M 的唯一极小厄布朗模型。

下面的定理表明这个术语的合理性，该定理说明 P 的稳定模型其实就是 P 的模型。

定理 7.9 P 的每个稳定模型都是 P 的极小模型。

证明 设 M 是 P 的稳定模型。考虑 P 的形如 $p:-q_1,\cdots,q_n,\neg s_1,\cdots,\neg s_m$ 的子句 C。如果某个 $s_j\in M$，那么 M 满足 C 是显然的。如果没有 s_j 在 M 中，那么 $p:-q_1,\cdots,q_n$ 就在 P_M 中。因为 M 是 P_M 的模型，所以当 $q_1,\cdots,q_n\in M$ 时，$p\in M$。因此在这个情形下 M 也满足 C，从而 M 是 P 的模型。

要证明 M 是 P 的极小模型，考虑 P 的任何模型 $M'\subseteq M$。我们需要证明 $M=M'$。根据稳定模型的定义，只需证明 M' 是 P_M 的模型。现在 P_M 中的任何子句 C' 来自 P 中如上面给出的某个子句 C，使得 $s_j\notin M$，其中 $1\leqslant j\leqslant m$。则 C' 形如 $p:-q_1,\cdots,q_n$。设 $q_1,\cdots,q_n\in M'$。证明 $p\in M'$。由于 $M'\subseteq M$，对于每个 $1\leqslant j\leqslant m$，$s_j\notin M'$。因此，由于 M' 是 $C\in P$ 的模型，所以 $p\in M'$，得证。 □

例 7.10 令 Q 为下面出自 Gelfond 和 Lifschitz[1988, 5.4]中的一般程序：

$$p(1,2).$$
$$q(x):-p(x,y),\neg q(y).$$

Q 有两个极小厄布朗模型：$M_1=\{p(1,2),q(1)\}$ 和 $M_2=\{p(1,2),q(2)\}$(习题 6)。将通常的因失败而否定规则运用于该程序，那么它对于问题“?-$q(2)$.”的回答是“no”，而对于问题“?-$q(1)$.”的回答是“yes”。因此我们更倾向于第一个模型。

现在考虑从厄布朗域的子集中选取该程序基本实例的稳定模型。首先，将程序本身转化为下面基本版的 P：

$$p(1,2).$$
$$q(1):-p(1,1),\neg q(1).$$

$$q(1) :\!- p(1,2), \neg q(2).$$
$$q(2) :\!- p(2,1), \neg q(1).$$
$$q(2) :\!- p(2,2), \neg q(2).$$

现在考虑厄布朗域的子集 $M = \{q(1)\}$。则 P_M 是

$$p(1,2).$$
$$q(1) :\!- p(1,2).$$
$$q(2) :\!- p(2,2).$$

P_M 的极小厄布朗模型是 $\{p(1, 2), q(1)\} \neq M$。因此，M 不是 P 的稳定模型。事实上，任何不含 $p(1, 2)$ 的集合 M 都不是 P 的模型，从而根据定理 7.9，它也不是 P 的稳定模型。

接着，考虑原来程序 Q 的两个极小厄布朗模型 M_1 和 M_2。我们断言 M_1 是稳定的而 M_2 不是。首先，P_{M_1}是

$$p(1,2).$$
$$q(1) :\!- p(1,2).$$
$$q(2) :\!- p(2,2).$$

这个程序的极小模型显然是 M_1，从而 M_1 是稳定的。另一方面，P_{M_2}是

$$p(1,2).$$
$$q(1) :\!- p(1,1).$$
$$q(2) :\!- p(2,1).$$

它的极小模型为 $\{p(1, 2)\} \neq M_2$。因此 M_2 不是稳定的。

事实上，M_1 是 P 仅有的稳定模型(习题 7)，从而根据因失败而否定的观点，稳定模型是“正确”的选择。

定理 7.12 给出稳定模型与非单调系统直接而精确的联系。我们继续使用上面的符号，并先从一个引理开始。

引理 7.11 *如果 M'是 P_M 的模型，那么 $M' \supseteq C_M(\varnothing)$。*

证明 设 M'为 P_M 的模型。通过对 M-演绎的长度进行归纳，证明 $C_M(\varnothing)$的每个成员都属于 M'。考虑任何(来自$\varnothing$的)M-演绎 $\varphi_1, \cdots, \varphi_k, p$，并设推出 p 的最后一步推导所用的规则是$\mathrm{tr}(C)$，其中 C 为 P 的子句。根据归纳，假设对于任何 $1 \leqslant i \leqslant k$, $\varphi_i \in M'$，从而$\mathrm{tr}(C)$的每个前件 q_i 都在 M'中。由于这是 M-演绎，$\mathrm{tr}(C)$的约束 s_j 都不在 M 中。则根据定义，$p :\!- q_1, \cdots, q_n$ 是 P_M 的子句之一。因为 M'是 P_M 的模型，所以有 $p \in M'$。 □

定理 7.12 *U 的子集 M 是 P 的稳定模型，当且仅当它是 $\mathrm{tr}(P)$的扩张。*

证明 设 M 是$\langle U, \mathrm{tr}(P)\rangle$的扩张。首先，我们断言 M 是 P_M 的模型。考虑 P_M 中的任何子句 $p :\!- q_1, \cdots, q_n$，其中 $q_1, \cdots, q_n \in M$。根据 P_M 的定义，P 中存在一个子句 $C = p :\!- q_1, \cdots, q_n, \neg s_1, \cdots, \neg s_m$，使得 s_j 不在 M 中。因此 $\mathrm{tr}(P)$中有一个规则 $\mathrm{tr}(C)$，其所有的前件在 M 中且没有约束在 M 中。因为根据命题 7.5，扩张是推理封闭的，所以 $q \in M$。其次，必须证明 M 的任何真子集 M'都不是 P_M 的模型。因为 $M = C_M(\varnothing)$，这可由引理 7.11 立即得到。

再考虑另外一个方向，设 M 是 P_M 的极小厄布朗模型。首先注意到，根据引理 7.11，

$M \supseteq C_M(\varnothing)$。根据 M 的极小性假设，只需证明 $C_M(\varnothing)$ 是 P_M 的模型就可得到所需要的结果 $M = C_M(\varnothing)$。因此，考虑 P_M 中任何子句 $p:-q_1, \cdots, q_n$，其中所有的 q_i 在 $C_M(\varnothing)$ 中。于是存在一个含所有 q_i 的 M -演绎 $\varphi_1, \cdots, \varphi_k$。根据 P_M 的定义，P 中存在一个子句 $C = p:-q_1, \cdots, q_n, \neg s_1, \cdots, \neg s_m$，使得 s_j 不在 M 中，从而在 tr(P)中有一个规则 tr(C)，使得其所有的前件在 $C_M(\varnothing)$ 中。由此可以构造一个以 p 为结果的 M -演绎。所以 $p \in C_M(\varnothing)$，得证。 □

Gelfond 和 Lifschitz 证明：具有第四节中所考虑的那些性质的一类程序有唯一的稳定模型，并且为这些程序提出稳定模型语义这个术语。唯一稳定模型这种特殊情形确实是特别有趣的。然而，对于非单调逻辑来说，tr(P)的所有扩张都可以作为系统模型的候选对象。

习题

1. 令 $S_1 \supseteq S_2 \supseteq \cdots$ 为单调系统 $\langle U, N\rangle$ 的推理封闭集的嵌套序列(nested sequence)。证明：$\cap S_i$ 是推理封闭的。
2. 佐恩(Zorn)引理(见第六章定理 10.2 和第十节习题 2)说，对于向下嵌套序列(downwardly nested sequence)的交封闭的非空集合类有极小元。应用这个引理和 1，证明：每个非单调形式系统有一个极小的推理封闭子集。
3. 证明：运算 $C_S(I)$ 对于 I 是单调的而对于 S 是反单调的，即，如果 $I \subseteq J$，那么 $C_S(I) \subseteq C_S(J)$，并且如果 $S \subseteq T$，那么 $C_S(I) \supseteq C_T(I)$。
4. 证明：如果 S 是 I 的扩张，那么 S 是 I 的极小推理封闭超集，并且对于任何满足 $I \subseteq J \subseteq S$ 的 J，有 $C_S(J) = S$。
5. 如果 S 和 T 是 I 的扩张且 $S \subseteq T$，那么 $S = T$。
6. 证明：例 7.10 中程序 P 和 Q 的极小厄布朗模型就是那里所给的集合 M_1 和 M_2。
7. 证明：在例 7.10 中 M_1 是 P 的唯一的稳定模型。(提示：在开始分析时应注意到，任何候选对象要成为 P 的模型必须含有 $p(1, 2)$，但是考虑到极小性，不可含有 P 的其他实例。)

   下面用到的图与偏序的基本术语可参考第一章第六节的习题 7～8。
8. 令 n 为自然数，G 为局部有穷图，即对该图的每个顶点 x，存在有穷多的顶点 y，使得 $\{x, y\}$ 是 G 的边。定义非单调的形式系统 $\langle U(G), N(G)\rangle$，使得 $U(G) = \{Cxi \mid x$ 是 G 的顶点且 $i \leqslant n\}$。然后，对于 G 的每个结点 x 和 $j \leqslant n$，把下面的规则放入 $N(G)$ 中：
$$\frac{:Cx1, \cdots, Cx(j-1), Cx(j+1), \cdots, Cxn}{Cxj}$$
   最后，对于 G 中每一对不同的顶点 x，y，每个 $i \leqslant n$ 和每个 $\varphi \in U(G)$，我们把下面的规则放入 $N(G)$ 中：
$$\frac{Cxi, Cyi}{\varphi}$$
   证明：如果把 Cxi 解释为顶点 x 着色 i，则 $S \subseteq U(G)$ 是 $\langle U(G), N(G)\rangle$ 的扩张，当且仅当 S 是 G 的一个 n -着色。
9. 令 P 是宽度为 n 的偏序。定义非单调系统 $\langle U(P), N(P)\rangle$，使得 $U(P) = \{Cxi \mid x \in P$ 且 $i \leqslant n\}$。对于每个 $x \in P$，向 $N(P)$ 中放入规则
$$\frac{:Cx1, \cdots, Cx(j-1), Cx(j+1), \cdots, Cxn}{Cxj}$$
   最后，对于在偏序 P 中不可比较的任何 x 与 y，向 $N(P)$ 中放入规则
$$\frac{Cxi, Cyi}{\varphi}$$
   证明：$S \subseteq U$ 是 $\langle U(P), N(P)\rangle$ 的扩张，当且仅当集合 $\{C_1, \cdots, C_n\}$ 是覆盖 P 的一些互不相交的链，其中 $C_i = \{x \mid C_{x_i} \in S\}$。

第八节　可计算性与不可判定性

20世纪三四十年代的逻辑学家的主要任务之一就是把算法或者能行方法(effective procedure)的直觉概念形式化。(为了方便我们考虑自然数上的算法。)研究者提出了许多看似不同的定义，他们包括丘奇、哥德尔(Gödel)、厄布朗、克林(Kleene)、马可夫(Markov)和波斯特(Post)。他们建议的方案有递归、等式演绎系统(equational deduction systems)、理想化计算机模型(idealized models of computing machines)等等。也许最有哲理说服力的是图灵(Turing)的建议。他所给出的一种简单的计算模型无疑是现在最著名的定义：图灵机(Turing machine)。

可以用这些模型计算的每个函数显然是能行的。随着研究的发展，人们发现任何直觉上可计算的函数都可以用这些系统中的任何一个去计算。事实上，在若干年之内所有这些建议被证明是相互等价的，就是说，用任何一种模型可以计算的函数类与用其他的模型可以计算的函数类是相同的。这些函数现在叫做**递归函数**(recursive function)。这些方面的结果导致丘奇给出了现在所说的**丘奇论题**(Church's thesis)：能行可计算的函数恰好就是递归函数。至今人们所获得的证据使得这个论题几乎被普遍接受。因此，要证明某个计算方案是通用的，即它可以计算每个能行的函数，只要证明：对于某个已知的定义了递归函数类的计算方案，它所能计算的每一个函数，我们的计算方案也能计算。

不难用谓词逻辑演绎对这些标准定义中的任何一个建立模型：对每个递归函数f，用算术语言写下一些公理，使得$f(n)=m$，当且仅当$p_f(\bar{n}, \bar{m})$是这些公理的逻辑后承，其中对每个自然数n，该算术语言含有一个项$\bar{n}$，且p_f为算术语言的一个二元谓词。(为了避免出现变元串，我们仅仅考虑一元函数。我们所做的每一件事同样适用于任意元的函数，只需用变元序列取代单个变元即可。)通常，这些表示可以自然地表达为PROLOG程序的形式。(以习题1~2为例。)因此，任何可靠且完全的PROLOG执行方法(例如，广度优先搜索)将可以正确地计算所有的递归函数。通过选取合适的计算模型(定义8.1中所描述的谢弗逊(Shepherdson)的寄存器机(register machine))并巧妙地把它翻译成PROLOG程序(定义8.4)，我们证明总选取最左边文字和对SLD-树进行深度优先搜索这个标准执行方法足以计算所有的递归函数。(其实，这些"可行的"程序本质上用任何选择规则和搜索方法都能正确地运行。)因此，PROLOG是一种通用的计算机模型(推论8.6和推论8.7)。

一旦人们对于算法或者能行可计算函数类有一个公认的数学定义，人们就有希望证明，一些方法(或判断)不能用任何算法实现(判定)，或者证明一些特别的函数不是能行可计算的。(这些概念其实是共存的。判断方法，例如判断一个数n是否在某给定集合A中，或者判断某个多项式是否有整数根，或者对应于计算A的特征函数C_A(当$n\in A$时，$C_A(n)=1$，当$n\notin A$时，$C_A(n)=0$)或者一个数组集合的特征函数，其中每个数组对应于一个有整数根的多项式的系数。)事实上，从递归函数的第一个定义开始直到目前，很多寻求算法的经典问题都被否定地解决了，即没有递归函数计算所期待的结果。其中最早也是最著名的结果之一是图灵对于停机问题(halting problem)不可判定性的证明：没有算法(递归函数)可以判断给定的计算机程序对于某个输入是否终止。因此，一旦把某种标准的计算模型翻译成PROLOG，也就证明了标准执行的PROLOG程序的停机问题是不可判定的。因为该论证适用于语义完全的执行，所以也证明了丘奇关于谓词逻辑永真性不可判定的著名结果(推论8.10)。

在开始证明 PROLOG 的计算通用性之前，我们给出谢弗逊关于可计算性的寄存器机模型。从“机械”上来说，它比图灵机简单，用 PROLOG 执行起来非常容易。寄存器机有一些存储单元称为寄存器。每个寄存器含有一个自然数。这些机器在运行一个程序时，只有两种操作可做。首先，它们可以把任何寄存器中的数增加 1 然后执行下一条指令。其次，它们可以检查给定的寄存器所含的数是不是 0。如果是，它们就执行下一条指令。如果不是，它们就把该寄存器中数减 1 然后执行指定的某个程序指令。我们定义寄存器机程序及其执行如下：

定义 8.1 **一个寄存器机程序**(register machine program)I 是对一个数列 x_1，…，x_r 进行操作的有穷指令序列 I_1，…，I_t，I_{t+1}，其中每个指令 $I_m(m\leqslant t)$是如下形式之一：

(i)$x_k:=x_k+1$(以 x_k+1 替换 x_k)。

(ii)If $x_k\neq 0$, then $x_k:=x_k-1$ and go to j.(如果 $x_k\neq 0$，把它替换为 x_k-1 并准备执行指令 I_j。)

约定，在执行某指令 I_m 后，应执行表列中下一条指令 I_{m+1}，除非 I_m 要求执行其他指令。对于输入值 x_1，…，x_r(寄存器最初含有的数值)，程序沿着指令序列按规定的方式执行各个指令，从而改变寄存器 x_k 的值。

最后一条指令 I_{t+1} 总是停机指令。因此，一旦到达 I_{t+1}，运行终止于寄存器的当前数值。一般地，用 $I_m(n_1, \cdots, n_r)$表示当前要执行的指令为 I_m 且当前各变元的值依次为 $n_1, \cdots, n_r$。

定义 8.2 寄存器机程序 I **计算一个函数** f: $\mathbb{N}\to\mathbb{N}$，如果当从指令 I_1 以及 $x_1=n$ 和 $x_k=0$(所有 $k>1$)开始时，程序运行最后终止于 $x_1=n$ 和 $x_2=f(n)$，即最终有 $I_{t+1}(n, f(n), n_3, \cdots, n_r)$，其中 n_3，…，n_r 为一些数值。如果 f 是从 $\mathbb{N}$ 到 $\mathbb{N}$ 的部分函数，我们还要求从 $I_1(n, 0, \cdots, 0)$开始执行的程序终止，当且仅当 n 属于 f 的定义域。

一个从 $\mathbb{N}$ 到 $\mathbb{N}$ 的(部分)函数是**(部分)递归的**，如果它可由某个寄存器机程序计算。

熟悉图灵机的读者可以证明，由寄存器机程序计算的部分函数恰好就是那些可由图灵机程序计算的部分函数。类似地，也可以考虑下面各种文献中给出的其他更一般的模型：Shepherdson 和 Sturgis[1963，3.6]与 Minsky[1961，3.6]等原创论文或者许多关于可计算性的基础教材，例如 Cutland[1980，3.6]或者 Tourlakis[1984，3.6]。对于我们的目的来说，我们仅用这一种模型来定义部分递归函数类。

为了确定性起见，我们给出具体的定义，说明一个 PROLOG 程序以某种特殊执行方式计算部分函数 f 的含义。设我们的语言可以表示一个极小的算术。事实上，我们所需要的仅仅是一个表示零的常元，比如 0，和一个一元函数 s，表示 $\mathbb{N}$ 上的后继函数。在这个背景中(我们在第二节的习题 7～8 中考虑过这个背景)，$s(x)$代表 $x+1$ 且 $s^n(0)$代表 n。

定义 8.3 一个带(二元)谓词 p 的 PROLOG 程序 P(用某种特殊的执行方法)**计算部分函数** f，如果对于自然数 a 与问题“?-$p(s^a(0), Z)$”，在 $f(a)$无定义情况下计算不停止，而在 $f(a)=b$ 的情况下计算停止并给出回答替换 $Z=s^b(0)$(没有其他答案)。如果没提到执行方式，我们默认采用任何某种既可靠又完全的执行方式(它们当然都是等价的)。所谓标准执行是指使用最左选择规则与深度优先搜索的运行方式，它是可靠的但不是完全的。

也许用谓词逻辑甚至 Horn 逻辑语言表示寄存器机的最自然方式，就是对于每个指令 I_m 引入一个有 r 个变量的谓词字母 p_m。而 $p_m(n_1, \cdots, n_r)$解释为，机器即将执行指令 I_m 且各变量的值为 n_1，…，n_r。现在可以很容易地用蕴涵关系把给定程序对应于指令的一步步执行

表示出来。

作为初次尝试，我们可以这样翻译：

对每个指令 I_m，$1\leqslant m\leqslant t$，使用如下某个恰当形式的公理：

(i)$p_m(x_1, \cdots, x_r)\rightarrow p_{m+1}(x_1, \cdots, x_{k-1}, s(x_k), x_{k+1}, \cdots, x_r)$。

(ii)$p_m(x_1, \cdots, x_{k-1}, 0, x_{k+1}, \cdots, x_r)\rightarrow p_{m+1}(x_1, \cdots, x_{k-1}, 0, x_{k+1}, \cdots, x_r)$

$\wedge p_m(x_1, \cdots, x_{k-1}, s(y), x_{k+1}, \cdots, x_r)\rightarrow p_j(x_1, \cdots, x_{k-1}, y, x_{k+1}, \cdots, x_r)$。

(注意，是某个自然数的后继等同于是非零的数。)

令 $Q(I)$ 为对应于寄存器机程序 I 的有穷的公理集。不难证明，如果程序 I 计算部分函数 f，那么对任何 n，m 和 $r-2$ 个数 a，b，$c\cdots$，$p_1(s^n(0), 0\cdots0)\rightarrow p_{t+1}(s^n(0), s^m(0), a, b, c\cdots)$ 是 $Q(I)$ 的逻辑后承，当且仅当 $f(n)$ 有定义且等于 m。这样的翻译足以表明所有递归函数可用任何既可靠又完全的 Horn 子句演绎系统或 PROLOG 进行计算，从而也得到该类系统的不可判定性(见习题 3)。遗憾的是，标准执行未必总能给出可以终止的运算(习题 4)。

(如 Bezem[1989，5.4]中所述)谢弗逊策略可以把寄存器机程序转化为(我们将看到甚至可以用标准运行方式)正确计算相同函数的 PROLOG 程序，它包括两个思想。其一是颠倒蕴涵的方向，因此把计算转变为验证。其二是转移目标子句中的变元，以防对它作不必要的合一尝试。

定义 8.4 *对应于寄存器机程序 I 的 PROLOG 程序 $P(I)$ 包含如下子句。*

转移目标子句中的变元并把控制交给第一个指令的子句：

$$p(X_1,Z) :- p_1(X_1,0,\cdots,0,Z).$$

(其中零串长度为 $r-1$。)

对应于每个(i)型指令 I_m 的子句：

$$p_m(X_1,\cdots,X_r,Z) :- p_{m+1}(X_1,\cdots,X_{k-1},s(X_k),X_{k+1},\cdots,X_r,Z).$$

对应于每个(ii)型指令 I_m 的两个子句：

$$p_m(X_1,\cdots,X_{k-1},0,X_{k+1},\cdots,X_r,Z) :-$$
$$p_{m+1}(X_1,\cdots,X_{k-1},0,X_{k+1},\cdots,X_r,Z)$$
$$p_m(X_1,\cdots,X_{k-1},s(Y),X_{k+1},\cdots,X_r,Z) :-$$
$$p_j(X_1,\cdots,X_{k-1},Y,X_{k+1},\cdots,X_r,Z).$$

最后，对应于寄存器机终止状态的子句：

$$p_{t+1}(X_1,X_2,\cdots,X_r,X_2).$$

定理 8.5 *对于任何寄存器机程序 I，带任何执行方式的 PROLOG 程序 $P(I)$ 与 I 计算相同的部分函数。*

证明 固定一个自然数 a 和计算部分递归函数 f 的寄存器机程序 I。考虑带目标子句 $G_0=\{\neg p(s^a(0), Z)\}$ 的程序 $P(I)$。首先注意到程序 $P(I)$ 中每个子句体至多有一个文字。因此，在任何由 G 开始的 SLD-证明中出现的每一个目标子句至多有一个文字。因此，任何这样的 SLD-证明过程与选择规则无关。我们还希望证明搜索 SLD-反驳的结果既是正确的又与搜索方法无关。要看清这一点需要更详细地分析程序的执行。

当从目标子句 $G_0=\{\neg p(s^a(0), Z)\}$ 开始时，第一个被消解的必定是 $P(I)$ 的第一个子句；结果为

$$G_1 = \{\neg p_1(s^a(0),0,\cdots,0,Z)\}$$

设寄存器机程序 I 的执行到第 n 步时状态为 $I_{m(n)}(m(1, n), m(2, n), \cdots, m(r, n))$，其中$m(n) \leqslant t+1$。我们断言：只要机器在第 n 步前没停止，该 SLD -证明的后继目标子句 G_n 一定是

$$\{\neg p_{m(n)}(s^{m(1,n)}(0),s^{m(2,n)}(0),\cdots,s^{m(r,n)}(0),Z)\}$$

且没有对 Z 做替换。用归纳法进行证明，因此设该断言对于 n 成立。

如果 $I_{m(n)}$是(i)型的，寄存器机的状态变为

$$I_{m(n)+1}(m(1,n),\cdots,m(k-1,n),m(k,n)+1,m(k+1,n),\cdots,m(r,n))$$

只有一个程序子句其头含 $p_{m(n)}$(对应于 $I_{m(n)}$)，从而只有一个程序子句可以与 G_n 消解。显然，这个消解可得所要求的结果：

$$\{\neg p_{m(n)+1}(s^{m(1,n)}(0),s^{m(2,n)}(0),\cdots,s^{m(k,n)+1}(0),\cdots,s^{m(r,n)}(0),Z)\}$$

如果 $I_{m(n)}$是(ii)型的，就有两个程序子句其头含 $p_{m(n)}$。根据是否 $m(k, n)=0$ 分情况论证。不管哪一种情况，恰好有一个子句其头可与 G_n 合一，所以该消解是唯一确定的且 G_{n+1} 有所要求的形式。

因此，如果从状态 $I_1(a, 0)$开始时 I 不终止(即 f 在 a 处无定义)，那么正如断言的那样，$P(I)$从目标$\{\neg p(s^a(0), Z)\}$开始时不终止。于是设$f(a)=b$。如果从状态 $I_1(a, 0)$开始，它一定在执行的第 $n+1$ 步停止于某个状态 $I_{t+1}(a, b, c_2, \cdots, c_r)$。根据上面的归纳论证，该(唯一的)SLD -证明中的子句 G_{n+1}是

$$\{\neg p_{t+1}(s^a(0),s^b(0),s^{c_2}(0),\cdots,s^{c_r}(0),Z)\}$$

再次注意到，恰好有一个程序子句其头含 p_{t+1}：该程序的最后一个子句。用该事实消解 G_{n+1} 可得□和一个含替换 $Z/s^b(0)$的 mgu，得证。 □

推论 8.6 每个(部分)递归函数 f 可以用一个 PROLOG 程序 P_f 计算，并且该程序可以用 PROLOG 的任何执行方式正确地运行。

证明 每个(部分)递归函数 f 可以用某个寄存器机程序 I_f 计算。根据上面的定理，$P(I_f)$是以任何执行方式计算f的 PROLOG 程序。 □

当然，我们提到 PROLOG 执行时，已经(默默地)假设有一个可以正确进行消解的执行方式(否则我们所说的就没有意义了)。特别地，与所有实际的执行方法相反，我们还必须假设一个正确的合一算法以确保任何结果的正确性。需要指出的是，即使省略合一算法中的出现检查，上面从寄存器机程序构造的 PROLOG 程序 $P(I)$也能正确地运行。

推论 8.7 每个(部分)递归函数 f 可以用一个 PROLOG 程序 P_f 计算，并且该程序可以用任何 PROLOG 执行方式正确地运行(即使省略合一算法中的出现检查)。

证明 像定理 8.5 证明中的分析那样，仔细观察从 $P(I)\cup\{G\}$出发由 $G=\{p(s^a(0), Z)\}$开始的 SLD -证明，可以看出所采用的替换都是基本的。在合一算法所作用的目标子句中，Z 一直是唯一的非基本项，在证明的最后一步之前，没有替换作用 Z。因此，这里针对程序子句中的变元所作的替换都是基本的。在最后一步(如果有的话)，对程序子句中的所有变元包括 Z 作一次基本替换。因为基本替换不会违背关于替换的出现条件，所以省略出现检查不会有什么影响。 □

现在可以证明 PROLOG 停机问题和谓词演算普遍永真性问题的不可判定性。

定理 8.8 寄存器机程序的停机问题是不可判定的，就是说，没有能行的方法可以判断

给定的某个程序对于某个输入是否终止。

证明 假设丘奇论题成立，即任何能行可计算的函数是递归的，也就是能用寄存器机程序计算。我们可以不用丘奇论题而形式地证明停机问题没有递归解，只要给下面论证中出现的少量的函数明确地写出计算它们的程序即可。这些程序可以在一些基本的论文或者许多关于可计算性的标准教材中找到。正如我们能列出所有的有穷数字序列，我们能列出寄存器机的所有程序。（见第六章定理 7.10 和定理 7.11。）假如停机问题是可判定的，就会有一个递归函数 $h(x)$，使得如果这个序列上第 x 个程序对于输入 x 终止且输出是 y，那么 $h(x)=y+1$，否则 $h(x)=0$。应用经典的对角线方法，立刻得到矛盾：因为（根据丘奇论题）h 是递归的，存在某个寄存器机程序计算 h。设序列上第 n 个程序计算 h。考虑 $h(n)$。由于 h 在每个数上有定义，第 n 个程序对于输入 n 终止并输出某个数 y。那么根据定义，$h(n)=y+1$，这与假设第 n 个程序计算 h 矛盾。 □

推论 8.9 使用任何执行方式的 PROLOG 程序，不管其合一算法带不带出现检查，其停机问题都是不可判定的。

证明 该证明直接由上面的定理以及定理 8.5 ~ 推论 8.7 的结果可得。 □

推论 8.10（丘奇定理） 谓词逻辑永真性问题是不可判定的，即没有能行的方法判断给定的谓词演算中的语句是否永真。

证明 如果 I 是一个寄存器机程序，那么根据定理 8.5，I 对于输入 a 终止，当且仅当寻找 $P(I)\cup\{\neg p(s^a(0), Z)\}$ 的 SLD－反驳的搜索成功地终止。根据 SLD－消解的完全性与可靠性，这种情况发生，当且仅当 $\exists Z\, p(s^a(0), Z)$ 是 $P(I)$ 的逻辑后承。因为 $P(I)$ 是一个有穷的子句集 $\{C_1, \cdots, C_n\}$，其终止等价于所有子句 C_i 全称闭包的合取蕴涵 $\exists Zp(s^a(0), Z)$。因此，假如我们能判定谓词演算的永真性问题，我们就能判定停机问题，这与定理 8.8 和推论 8.9 矛盾。 □

习题

部分递归函数（partial recursive function）有一个很常见的定义是用复合、原始递归和极小化运算作用于几个简单函数而得：

（1）后继函数 $s(x)=x+1$ 是部分递归的。

（2）常函数 $c(x)=0$ 是部分递归的。

（3）对于每个 i 和 r，投影函数 $p_{i,r}(x_1, \cdots, x_r)=x_i$ 是部分递归的。

（4）如果 $g_1, \cdots, g_n$ 和 h 是部分递归的，那么 $f(x_1, \cdots, x_r)=h(g_1(x_1, \cdots, x_r), \cdots, g_n(x_1, \cdots, x_r))$ 也是部分递归的。

（5）如果 $r>1$ 且 g 和 h 是部分递归的，那么如下“用原始递归”定义的函数 f 也是部分递归的：

$$f(0,x_2,\cdots,x_r)=g(x_2,\cdots,x_r)$$

$$f(x_1+1,x_2,\cdots,x_r)=h(x_1,f(x_1,x_2,\cdots,x_r),x_2,\cdots,x_r)。$$

（6）如果 $f(x_1, \cdots, x_r, y)$ 是部分递归的，那么函数 $g(x_1, \cdots, x_r)$ 也是部分递归的，其中 $g(x_1, \cdots, x_r)$ 等于最小数 y，使得 $f(x_1, \cdots, x_r, z)$ 对于每个 $z\leqslant y$ 都有定义且 $f(x_1, \cdots, x_r, y)=0$。

1. （对上面的定义用归纳法）说明如何写出计算某个部分递归函数的 PROLOG 程序。
2. 证明：习题 1 答案给出的所有程序用任何可靠而完全的 PROLOG 执行都可以正确地运算。（提示：根据 PROLOG 执行的完全性，只需对每个计算找到一个 LD－反驳。现在沿着计算路径进行。对于另一个方向，应用可靠性并注意到对于这些程序本来的语义而言，N 是一个自然的模型。）
3. 如同定义 8.4 之前所给出的那样，令 $Q(I)$ 为对应于计算部分递归函数 f 的寄存器机程序 I 的子句集。

(i)从语义上论证，对任何数 n，m，a，b，c，…，

$$p_i(s^n(0),0,\cdots,0)\rightarrow p_{i+1}(s^n(0),s^m(0),s^a(0),s^b(0),\cdots)$$

是 $Q(I)$ 的逻辑后承，当且仅当 $f(n)$ 有定义且等于 m。

(ii)应用定理 8.8 给出定理 8.10 的证明，使得该证明不涉及 PROLOG 执行或者消解等概念。

(iii)证明：$Q(I)$ 作为 PROLOG 程序可以用任何可靠且完全的执行正确地计算寄存器机 I 所计算的函数。(提示：见习题 2 的提示。)

4. 举一个寄存器机程序 I 的例子，使得 $Q(I)$ 作为 PROLOG 程序用标准执行不能正确地计算 I 所计算的函数。

5. 一个自然数集 W 是递归可枚举的(recursively enumerable)，如果有一个能行的方法列出其成员，即它是空集或者是某个(全)递归函数的值域。用丘奇论题证明：集合 W 是递归的，当且仅当 W 与其补集 $N-W$ 都是递归可枚举的。(提示：非形式地论证，给定 W 与 $N-W$ 的枚举序列，可以计算 W 的特征函数。)

6. 证明：存在一个 PROLOG 程序 P，使得第六节中所定义的 CWA(P)的逻辑后承不是递归可枚举的。(提示：根据丘奇论题和本节的结果，存在一个计算部分递归函数 f 的程序 P，其中 $f(n,x)=0$，当且仅当第 n 个寄存器机程序对于输入 x 终止。因为 CWA(P)唯一的模型是N，CWA(P)的逻辑后承恰好是有关这个函数的真事实。特别地，$\neg p(s^n(0),0)$ 是 CWA(P)的逻辑后承，当且仅当第 n 个寄存器机程序对于输入 x 不终止。再应用习题 5 说明这与定理 8.8 矛盾。)

进一步阅读建议

Apt[1990，5.4]是一篇很好的关于逻辑式编程的综述文章。1992 年三月份那一期的《Communications of the ACM》专门讨论逻辑式编程。其中有一些有趣的文章包括一篇简史(Robinson[1992，5.4])。PROLOG 理论的标准教材是 Lloyd[1987，5.4]，其中给出了大量的参考书目。另外一本侧重于消解以及逻辑式编程与关系数据库的联系的教材是 Maier 和 Warren[1988，5.4]。最近的两本教材是 Doets[1994，5.4]和 Apt[1996，5.4]。

PROLOG 编程实践见 Bratko[1970，5.4]、Sterling 和 Shapiro[1986，5.4]、O'Keefe[1990，5.4]或者 Sterling[1990，5.4]。Clocksin 和 Mellish[1987，5.4]曾经是标准的教材，现在仍然适用。如果想了解法国人和日本人的观点，可以分别参考译著 Boizumault[1995，5.4]和 Mizoguchi[1991，5.4]。

要进一步了解逻辑式编程与 PROLOG 的终止问题，见 Apt 和 Pedreschi[1991，5.4]以及所引用的参考文献。

带等号的逻辑可以参考第一章后的进一步阅读建议中所提到的所有关于数理逻辑的标准教材。可以首先考虑 Mendelson[1979，3.2]、Enderton[1972，3.2]或 Shoenfield[1967，3.2]。关于模型论基础的书如 Chang 与 Keisler[1973，3.4]也值得看一看。

有关 PROLOG 中因失败而否定规则的论述见 Lloyd[1987，5.4]的第三章和第四章以及所引用的参考文献。在 Shepherdson[1992，5.4]中可以发现一个很好的综述以及大量的参考书目。

关于非单调逻辑的基础性文章见 Ginsberg[1987，5.5]。要了解第七节中方法的进一步发展，我们推荐 Marek、Remmel 和 Nerode[1990 和 1993，5.5]。要了解第七节开头提到的非单调逻辑的不同的研究方法及起源，见 Apt、Blair 和 Walker[1988，5.4]、Clark[1978，5.4]、Doyle[1979，5.5]、Hintikka[1962，5.5]、McCarthy[1980，5.5]、Minsky[1975，5.5]、Moore[1985，5.5]和 Reiter[1980，5.5]。当前研究的论文一个很好的来源是表 5.5 中列出的几卷 TARK(《Theoretical Aspects of Reasoning about Knowledge》)。Marek 和

Truszczynski[1993，5.5]是关于这一学科的第一本真正的数学教材。

要看一本很棒的承认丘奇论题并且从非形式观点介绍递归函数理论(可计算性理论)的书，我们推荐 Rogers[1967，3.6]。要了解几种不同的定义相互等价的细节及其具体“做法”，我们推荐 Davis 和 Weyuker[1983，5.2]及 Odifreddi[1989，3.6]。想进一步专门了解寄存器机，见 Fitting[1987，5.2]、Cutland[1980，3.6]、Tourlakis [1984，3.2]或者 Shepherdson 与 Sturgis 的开创性论文[1963，3.6]和 Minsky[1961，3.6]。Machtey 和 Young [1978，5.4]中有一些更加面向计算机科学的研究。

沿着谓词逻辑不可判定性的证明思路，接着出现的是著名的哥德尔不完全性定理：一阶算术谓词逻辑语言中存在语句 φ，使得 φ 与 $\neg\varphi$ 都不能用任何合理的算术公理证明。Crossley 等人的[1990，3.2]是关于不完全性结果的简要的入门教材。这个结果在许多标准的教材中也可以见到，例如 Boolos 和 Jeffrey[1989，3.2]，Enderton[1972，3.2]和 Mendelson[1979，3.2]。以递归论的观点所做的处理见 Odifreddi[1989，3.6]。对于难题的研究(a puzzler's approach)可见 Smullyan[1987 和 1978，3.2]。Gödel 的原创论文[1931，2.3]仍然值得一读。

第四章 模态逻辑

第一节 可能性与必然性；知识或信念

发展形式模态逻辑的目的是为了准确地表达一些概念之间不同的数学性质，比如哲学和自然语言中的可能性、必然性、信念、知识和时态发展等概念。自从20世纪70年代以来，模态逻辑已经成为计算机科学和人工智能中表达基本观念的有用工具。

从形式上来说，模态逻辑是经典的命题逻辑或谓词逻辑的扩展。在经典逻辑语言上添加了新的“模态算子”。标准的基本算子传统上表示为$\Box$和$\Diamond$。语法上来说，它们可以视为新的一元联结词。(我们省略对于命题逻辑的单独处理，直接面向谓词逻辑。正如第二章定理4.8中经典逻辑那样，命题逻辑可以视为谓词逻辑的子集，所以关于谓词逻辑的结果包含关于命题逻辑的结果。经典逻辑与模态逻辑之间也具有这样的关系。)

定义 1.1 如果$\mathcal{L}$是(经典的)谓词逻辑语言(如第二章第二节中的定义)，那么(在第二章定义2.1中)添加两个新的原始符号$\Box$和$\Diamond$，把它扩展为**模态语言**(modal language)$\mathcal{L}_{\Box,\Diamond}$。对于公式的定义(第二章定义2.5)我们加上一个新的条款：

(iv)如果φ是一个公式，那么$(\Box\varphi)$和$(\Diamond\varphi)$也是。

其他如子公式、约束变元和语句等有关概念可以一字不差地移植过来。

在不引起混淆时，$\mathcal{L}_{\Box,\Diamond}$中的脚标可以省去，仅记为(模态)语言$\mathcal{L}$。

对于模态语言的解释最初是哲学所考虑的问题。$\Box$和$\Diamond$一般读作“……是必然的”与“……是可能的”。另外一种读法是“……将总是真的”和“……将最终是真的”。你也许注意到$\Box$和$\Diamond$的关系类似于$\forall$和$\exists$的关系。它们是下面意义下的对偶算子，即$\Diamond\varphi$通常的含义是$\neg\Box\neg\varphi$。根据刚才提到的两种解释，这两个算子各有一个常用的名字。有时候只需要使用一个算子。例如，在有关知识或信念的解释中，往往只用算子$\Box$(这样的语言记为$\mathcal{L}_{\Box}$)，且$\Box\varphi$理解为“我知道φ”或“我相信φ”。也可能使用更多的模态算子$\Box_i$和$\Diamond_i$并给以不同的解释。我们倾向于把$\Box$和$\Diamond$读作“方框”和“菱形”，这样可以避免预先确定其解释。

模态语言$\mathcal{L}_{\Box,\Diamond}$的语义以Kripke框架(Kripke frame)为基础，它是第二章第四节谓词逻辑语义中结构这个概念的推广。从直观上来说，我们考虑一个“可能世界”(possible world)族W。每个世界$w\in W$代表着一种对于现实的看法，用结构$\mathcal{C}(w)$表示。我们借用集合论的力迫(forcing)符号，用$w\Vdash\varphi$表示φ在可能世界w中为真。(我们把$w\Vdash\varphi$读作“w力迫φ”或“φ在w中为真”。)如果φ是经典语言$\mathcal{L}$的语句，这应该理解为φ在结构$\mathcal{C}(w)$中为真。如果$\Box$解释为必然，那么我们把必然这个概念理解为在所有可能世界中为真；用$\Diamond$所表示的可能性概念，其意思是在某个可能世界中为真。

时态概念或关于某个事实φ在某些事件已存在情况下的必然性或可能性的断言，可以用可能世界之间的可达(或者后继)关系S来表示。因此，我们用$w\Vdash\Box\varphi$表示φ在w的所有可能后继世界或w的所有可达世界中为真。这是“φ在世界w中必然为真”的一个合理的解释。

在第二节形式化模态逻辑的语义之前，我们先看看计算机科学中的两种应用，从中可以

看出模态逻辑的另外一个动机。

第一个应用领域是程序行为理论。在 Turing[1949, 5.7]、Von Neumann[1961, 5.7]、Floyd[1967, 5.7]、Hoare[1969, 5.7]和 Burstall[1972, 5.7]等关于程序正确性的著作中，模态还没有明确的定义，其内在的模态逻辑系统是许多后来人揭示出来的。最近由于程序分析而发展起来的逻辑包括算法逻辑(algorithmic logic)、动态逻辑(dynamic logic)、进程逻辑(process logic)和时态逻辑(temporal logic)。下面给出其中一个逻辑系统即顺序程序(sequential program)的动态逻辑所使用的模态词。

令 α 为一(可能是非确定的)顺序程序，并令 s 为执行 α 的机器的一个状态。令 φ 为关于状态的谓词或者性质。我们引入模态算子$\Box_\alpha$ 和$\Diamond_\alpha$ 来描述程序 α 的执行，其中$\Box_\alpha\varphi$ 解释为在 α 执行后 φ 必然为真或总是为真。$\Diamond_\alpha$ 的意思是在 α 执行时 φ 有时为真(即 α 的某个执行使 φ 为真。)因此$\Box_\alpha$ 是模态必然性算子，$\Diamond_\alpha$ 是模态可能性算子。

采用上面描述的可能世界观点可以使得这个语言的作用更大。这里“可能世界”是机器的状态，而可达关系由程序 α 的可能执行序列确定。更准确地，我们对模态公式的力迫性断言作如下解释：

$s \Vdash \Box_\alpha\varphi$ 断言：φ 在任何状态 s'下为真，使得程序 α 有一个从 s 开始并最终到达 s'的合法的执行。

$s \Vdash \Diamond_\alpha\varphi$ 断言：φ 在某个状态 s'下为真，使得程序 α(至少)有一个从 s 开始并最终到达 s'的合法的执行。

因此，α 所确定的可达关系 S_α 是指从 s 可达 s'，记为 $sS_\alpha s'$，当且仅当从状态 s 开始执行程序 α 最终可以到达状态 s'。

对于每个程序 α 可以引入不同的算子$\Box_\alpha$ 和$\Diamond_\alpha$。这样就可以对于每一对算子$\Box_\alpha$ 和$\Diamond_\alpha$ 用可达关系 S_α 形成模态克里普克语义。这样的语言在讨论程序的等价变形与证明它们的正确性时是非常有用的。毕竟，其正确性不过是断言：不管开始状态是什么，当程序 α 执行结束时，某个情形 φ 总是为真：$\Box_\alpha\varphi$。(见 Goldblatt[1982, 5.6]、[1992, 5.6]和 Harel[1979, 5.7]。)

这个主题有许多有趣和有用的变体。例如，可以把 $s \Vdash \Box_\alpha\varphi$ 解释为在 α 从 s 开始执行所到达的每一个状态 s'上 φ 都为真。相应地，$s \Vdash \Diamond_\alpha\varphi$ 指 φ 在 α 从 s 开始执行所到达的某一个状态 s'上 φ 为真。我们只要改变一下可达关系便得到所谓的进程逻辑。这个解释与时态逻辑也很接近。在时态逻辑中，$\Box\varphi$ 的意思是 φ 总为真，而$\Diamond\varphi$ 的意思是 φ 最终(或有时候)为真。这一逻辑对于不同时间观念的模态算子可以有不同的扩展。在一个数字时序机(digital sequential machine)中，把时间视为自然数那样的序列也许是合理的。在这种情形下，举例来说，可以引入模态算子$\circ$并把$t \Vdash \circ\varphi$读作 φ 在 t 的直接后继时刻为真。再举一个例子，可以用这些系统(甚至可以不用$\circ$)表达不同的公平观念：每个持续活跃的过程终将在计划安排之中(被执行或审查等等)——$\Box\varphi(c)\to\Diamond\psi(c)$；每个曾经活跃的过程至少一次被列入计划安排之中——$\varphi(c)\to\Diamond\psi(c)$；每个无穷次活跃的过程将被无穷次地列入计划——$\Box\Diamond\varphi(c)\to\Box\Diamond\psi(c)$；等等。因此这些逻辑可用于分析并行程序(concurrent program)或不间断程序(perpetual program)的一般行为，特别是它们的正确性。(这里另外一本很好的参考书是 Manna 和 Waldinger[1985, 5.6]。)

模态逻辑在计算机科学中的一个十分不同的应用领域是人工智能知识与信念理论。这里

我们可以把$\Box_K\varphi$理解为“某个(固定的)代理或处理器知道φ(即φ为真)”，或者把$\Box_B\varphi$理解为“某个(固定的)代理或处理器相信φ(即φ为真)”。我们再次希望讨论的不仅仅是一个而是很多的处理器。于是引入这样一些模态算子如$\Box_{K,\alpha}\varphi$，把它理解为“处理器α知道φ”。因此，例如，$\Box_{K,\alpha}\Box_{K,\beta}\varphi$的意思是：$\alpha$知道$\beta$知道$\varphi$；$\Box_{K,\alpha}\varphi\rightarrow\Box_{K,\beta}\psi$的意思是：如果$\alpha$知道$\varphi$，那么$\beta$知道$\psi$。(关于人工智能逻辑，一般参考 Turner[1984，5.6]。)

这一语言显然允许我们表述分布式或并发系统中关于通信和知识的观念。另外一种相关的研究道路考虑把人类以及机器系统的信念或知识公理化。这样可以从公理中推出知识或信念的其他性质。根据这些推理，人们也许要么改变自己的认知观，要么改变自己所乐于接受的关于知识的公理。把模态逻辑视为信念或知识的逻辑的观点与数据库管理分析密切相关。因而，它也与第三章第七节的非单调逻辑密切相关。(见 Fagin 等人的[1995，5.6]中关于知识与信念逻辑的综述和 Thayse[1989，5.6]中对于应用于演绎数据库和人工智能的模态逻辑的详细论述。)

在下面几节中给出模态逻辑的形式语义(第二节)和一个图表式证明系统(第三节)。在第四节中证明了这个证明系统的可靠性和完全性定理。模态逻辑的许多应用都针对这样的一些系统，其中允许(或建议)对有关的解释加上一些限制，以反映我们对于必然性、知识、时间等概念的不同观点。在第五节中探讨可达关系上不同限制之间的关系，对其内在逻辑添加模态算子公理，并给出新的表证明规则。最后一节(第六节)描述一个传统的希尔伯特式模态逻辑系统，它扩展了第二章第八节的经典逻辑。

第二节　框架和力迫

为了技术上的方便，我们对基本(模态)语言$\mathcal{L}$作两个修改。首先，把联结词$\leftrightarrow$从我们的形式语言中省略掉，仅把$\varphi\leftrightarrow\psi$视为$\varphi\rightarrow\psi\wedge\psi\rightarrow\varphi$的缩写。其次，设本章中每个语言$\mathcal{L}$至少有一个常元符号，并且除了常函数之外没有其他的函数符号。(删除函数符号不会严重损害语言的表达能力。我们可以全部用关系代替函数符号。例如，二元函数符号$f(x, y)$的功能可以用三元关系符号$R_f(x, y, z)$代替，其意思是$f(x, y)=z$。从而公式$\varphi(f(x, y))$也可全部替换为$\exists z(R_f(x, y, z)\wedge\varphi(z))$。)

现在给出表示模态逻辑语义的框架的精确概念。正如我们曾经解释的，框架由一个“可能世界”集合W、可能世界之间的可达(或后继)关系S和给每个$p\in W$赋予一个经典结构$\mathcal{C}(p)$的指派这三个部分所构成。我们要求结构$\mathcal{C}(p)$的论域$C(p)$对于后继关系是单调的，即如果q是p的后继世界，记为pSq，那么$C(p)\subseteq C(q)$。这个弱单调性要求并不是一个严格的限制。因为即便是原子谓词也并没有被要求具有单调性，即$C(p)$的元素c可以在$C(p)$中有性质R而在$C(q)$中没有，所以我们可以宣称任何对象不再是某个特殊数据库或者谓词的论域中的元素。人们也可以给出没有这个限制的框架语义，但这样会出现许多我们希望避免的困难。例如，如果论域中没有任何对象，即有某个$C(q)=\varnothing$，那么我们就完全脱离了经典谓词逻辑的领域，因为谓词逻辑所探讨的论域都是非空的。

定义 2.1　令$\mathcal{C}=(W, S, \{\mathcal{C}(p)\}_{p\in W})$由集合$W$、$W$上的二元关系$S$和一个给$W$中每个$p$分配语言$\mathcal{L}$的(在第二章定义 4.1 意义下经典的)一个结构$\mathcal{C}(p)$的函数构成。为简化符号，我们把$\mathcal{C}=(W, S, \{\mathcal{C}(p)\}_{p\in W})$记为$\mathcal{C}=(W, S, \mathcal{C}(p))$。像通常一样，用$C(p)$表示结构$\mathcal{C}(p)$的论域。以第二章第四节的真值定义中那样的方式，向$\mathcal{L}$添加$c_a$作为$C(p)$的每个元素

a 的名字，所得的扩张记为 $\mathcal{L}(p)$。用 pSq 或 $(p, q) \in S$ 表示关系 S 在 p 和 q 之间成立。我们也把这一状态描述为从 p **可达**(accessible)q(或者 q 是 p 的**后继**)。我们称$\mathcal{C}$是语言$\mathcal{L}$的一个**框架**(frame)，或者简称为$\mathcal{L}$**框架**，如果对于 W 中每个 p 和 q，pSq 蕴涵 $C(p) \subseteq C(q)$ 且$\mathcal{L}(p) \subseteq \mathcal{L}(q)$中常元的解释与其在$\mathcal{C}(p)$以及$\mathcal{C}(q)$中的解释相同。

现在给$\mathcal{L}$框架定义力迫关系。在阅读下面定义和后面例子的时候，记住下面的解释范例是有用的：每个 $p \in W$ 是一个可能世界；pSq 意思是 q 是 p 的一个可能未来；$p \Vdash \varphi$ 意思是 φ 在世界 p 中为真；$\Box\varphi$ 意思是 φ 将一直为真，而$\Diamond\varphi$ 的意思是 φ 在未来某个时候为真。

定义 2.2(框架的力迫)　令$\mathcal{C} = (W, S, C(p))$为语言$\mathcal{L}$的一个框架，$p$ 在 W 中且 φ 是语言$\mathcal{L}(p)$的语句。对语句 φ 作归纳，我们定义 p **力迫**(force)φ，记为 $p \Vdash \varphi$。

(i)对于原子语句 φ，$p \Vdash \varphi \Leftrightarrow \varphi$ 在$\mathcal{C}(p)$中为真。

(ii)$p \Vdash (\varphi \rightarrow \psi) \Leftrightarrow p \Vdash \varphi$ 蕴涵 $p \Vdash \psi$。

(iii)$p \Vdash \neg\varphi \Leftrightarrow p$ 不力迫 φ(记为 $p \nVdash \varphi$)。

(iv)$p \Vdash (\forall x)\varphi(x) \Leftrightarrow$ 对于$\mathcal{L}(p)$中的每个常元 c，$p \Vdash \varphi(c)$。

(v)$p \Vdash (\exists x)\varphi(x) \Leftrightarrow \mathcal{L}(p)$中存在一个常元 c，使得 $p \Vdash \varphi(c)$。

(vi)$p \Vdash (\varphi \wedge \psi) \Leftrightarrow p \Vdash \varphi$ 且 $p \Vdash \psi$。

(vii)$p \Vdash (\varphi \vee \psi) \Leftrightarrow p \Vdash \varphi$ 或 $p \Vdash \psi$。

(viii)$p \Vdash \Box\varphi \Leftrightarrow$ 对于任何使得 pSq 的 $q \in W$，$q \Vdash \varphi$。

(ix)$p \Vdash \Diamond\varphi \Leftrightarrow$ 存在 $q \in W$，使得 pSq 且 $q \Vdash \varphi$。

如果要指明所用框架，我们可以说在$\mathcal{C}$中 p 力迫 φ，并记为 $p \Vdash_{\mathcal{C}} \varphi$。

定义 2.3　令 φ 为语言$\mathcal{L}$的语句。我们说 φ 在$\mathcal{L}$**框架**$\mathcal{C}$**中被力迫**，记为$\Vdash_{\mathcal{C}}\varphi$，如果 W 中的任何 p 力迫 φ。我们说 φ 是**永真的**，记为$\vDash\varphi$，如果 φ 在每个$\mathcal{L}$框架$\mathcal{C}$中被力迫。

例 2.4　对于任何语句 φ，语句$\Box\varphi \rightarrow \neg\Diamond\neg\varphi$ 是永真的：考虑任何框架$\mathcal{C} = (W, S, C(p))$和任何 $p \in W$。我们必须根据定义 2.2 的(ii)验证 $p \Vdash \Box\varphi \rightarrow \neg\Diamond\neg\varphi$。于是设 $p \Vdash \Box\varphi$。如果 $p \nVdash \neg\Diamond\neg\varphi$,那么(根据(iii))$p \Vdash \Diamond\neg\varphi$。根据(ix)，存在 $q \in W$，使得 pSq 且 $q \Vdash \neg\varphi$。由假设 $p \Vdash \Box\varphi$ 与(viii)可知 $q \Vdash \varphi$，与(iii)矛盾。习题 1 证明逆命题$\neg\Diamond\neg\varphi \rightarrow \Box\varphi$ 也是永真的。

例 2.5　我们断言：$\Box\forall x\varphi(x) \rightarrow \forall x\Box\varphi(x)$是永真的。如果不是，则存在框架$\mathcal{C}$和 p，使得 $p \Vdash \Box\forall x\varphi(x)$但 $p \nVdash \forall x\Box\varphi(x)$。如果 $p \nVdash \forall x\Box\varphi(x)$，那么根据(iv)存在 $c \in \mathcal{L}(p)$，使得$p \nVdash \Box\varphi(c)$。然后根据(ix)，存在 $q \in W$ 使得 pSq 且 $q \nVdash \varphi(c)$。由于 $p \Vdash \Box\forall x\varphi(x)$，根据(ix)有$q \Vdash \forall x\varphi(x)$。最后，根据(iv)有 $q \Vdash \varphi(c)$，由此得出矛盾。注意，论域 $C(p)$的单调性假设，即 $pSq \Rightarrow C(p) \subseteq C(q)$，在论证中起着关键作用。

例 2.6　$\Box\varphi(c) \rightarrow \varphi(c)$不是永真的：考虑任何这样的框架，其中原子语句 $\varphi(c)$在某个$\mathcal{C}(p)$中不真且没有 q 使得 pSq。在这样的框架中，$p \Vdash \Box\varphi(c)$但 $p \nVdash \varphi(c)$。

例 2.7　$\forall x\varphi(x) \rightarrow \Box\forall x\varphi(x)$不是永真的：令$\mathcal{C}$为一框架，其中 $W = \{p, q\}$，$S = \{(p, q)\}$，$C(p) = \{c\}$，$C(q) = \{c, d\}$，$\mathcal{C}(p) \vDash \varphi(c)$且$\mathcal{C}(q) \vDash \varphi(c) \wedge \neg\varphi(d)$。现在 $p \Vdash \forall x\varphi(x)$，但 $p \nVdash \Box\forall x\varphi(x)$，因为 $q \nVdash \varphi(d)$。这个例子的关键在于没有设框架$\mathcal{C}$的各论域 $C(p)$相同。第四节习题 8 考虑了限制在常论域上的模态逻辑。

注意对于不含模态算子的语句 φ 而言，这里所定义的永真性与经典谓词逻辑的定义相同(习题 10)。

在定义模态逻辑"逻辑后承"时应当小心一点。如果基本结构是整个框架而不是其中单独的世界，那么就会得出下面的定义：

定义 2.8 令 Σ 为模态语言 $\mathcal{L}$ 的一个语句集而 φ 是 $\mathcal{L}$ 的一个语句。φ 是 Σ 的**一个逻辑后承**，记为 $\Sigma \vDash \varphi$，如果当每一个 $\psi \in \Sigma$ 都在某个 $\mathcal{L}$ 框架 $\mathcal{C}$ 中被力迫时，φ 也在这个框架中被力迫。

提醒 这个逻辑后承定义没有要求，在每个 $\mathcal{L}$ 框架 $\mathcal{C}$ 中，φ 在每个使得所有 $\psi \in \Sigma$ 为真的世界 w 中为真(习题 11)。特别地，演绎定理(第二章第七节习题 6)对于模态逻辑不成立，这可从例 2.7 和例 2.9 看出来。

例 2.9 $\forall x\varphi(x) \vDash \Box\forall x\varphi(x)$：设 $\mathcal{C}$ 是一框架，使得对于每个可能世界 $p \in W$，有 $p \Vdash \forall x\varphi(x)$。如果 $q \in W$，我们断言 $q \Vdash \Box\forall x\varphi(x)$。否则存在 $p \in W$，使得 qSp 且 $p \nVdash \forall x\varphi(x)$，与我们的假设矛盾。

例 2.10 如果 φ 是一元原子谓词，$\Box\varphi(c) \nvDash \Diamond\varphi(c)$：考虑框架 $\mathcal{C}$，其中 $S = \varnothing$，且 $\mathcal{C}(p) \nvDash \varphi(c)$，从而对于每个 p，有 $p \nVdash \varphi(c)$。在 $\mathcal{C}$ 中，每个 p 力迫 $\Box\varphi(c)$，但没有一个力迫 $\varphi(c)$，从而没有一个力迫 $\Diamond\varphi(c)$。

对可能世界集合 W 或(更常见地)对可达关系 S 作限制还可以得到其他的永真性概念(从而得到其他的逻辑后承概念)。例如，仅考虑自反且传递的可达关系常常是有用的。在第五节会讨论其中的几个结果。

我们应当意识到尽管 $\Box$ 和 $\Diamond$ 在语法上是命题联结词，但它们的语义涉及所有可能世界。$\Box\varphi$ 是说，不管你到达哪个后继世界，φ 都在那里为真。$\Diamond\varphi$ 是说，存在一个后继世界，你可以到达那里并使得 φ 为真。对于这样的语义构造其表，自然涉及新世界的引入和对旧世界元素进行实例化。

习题

在定义 2.3 中永真性的语义定义基础上，证明以下是永真的模态语句。

1. $\neg\Diamond\neg\varphi \to \Box\varphi$(对任何语句 φ)

2. $\forall x\Box\varphi(x) \to \exists x\Box\varphi(x)$(对任何只有一个自由变元 x 的公式 $\varphi(x)$)

证明下面的模态语句一般来说不是永真的。令 φ 为任一模态语句。

3. $\varphi \to \Diamond\varphi$

4. $\varphi \to \Box\varphi$

5. $\Diamond\varphi \to \varphi$

验证模态语句 φ 的以下逻辑后承实例：

6. $\varphi \vDash \Box\varphi$

7. $(\varphi \to \Box\varphi) \vDash (\Box\varphi \to \Box\Box\varphi)$

给出框架表明下面的逻辑后承不成立：

8. $\Box\varphi \nvDash \varphi$

9. $(\Box\varphi \to \varphi) \nvDash (\Box\varphi \to \Box\Box\varphi)$

10. 如果 φ 是一个不含 $\Box$ 或 $\Diamond$ 的语句，证明：φ 在定义 3.1 意义下的永真性与在第二章定义 4.4 意义下的永真性相同。

11. 我们说 φ 是 Σ 的局部后承(local consequence of Σ)，如果对于每个 $\mathcal{L}$ 框架 $\mathcal{C} = (W, S, \mathcal{C}(p))$，$\forall p \in W[(\forall\psi \in \Sigma)(p \Vdash \psi) \to p \Vdash \varphi]$。

(i)证明：如果 φ 是 Σ 的局部后承，那么它是 Σ 的逻辑后承。

(ii)证明：(i)的逆不成立，即 φ 可以是 Σ 的逻辑后承但不是局部后承。

第三节 模 态 表

我们描述模态逻辑的一种证明方法，它以一个类似于第二章第六节中经典逻辑的表系统为基础。在经典逻辑中，进行表证明的策略是系统地搜索与起始标号语句一致的结构。我们或者找到这样的结构，或者看到每个可能的分析都导致矛盾。当从某个标号语句 $F\varphi$ 开始时，要么找到一个结构，φ 在其中为假，要么断定有 φ 的一个证明。对于模态逻辑我们从标号力迫断言(signed forcing assertion) $Tp \Vdash \varphi$ 或 $Fp \Vdash \varphi$(φ 是一个语句)开始尝试，建立一个与该断言一致的框架，或者断定任何这样的尝试都将导致矛盾。如果从 $Fp \Vdash \varphi$ 开始，那么要么找到一个框架，在其中 p 不力迫 φ，要么断定有 φ 的一个模态证明。

模态逻辑的表和表证明的定义，在形式上都很像第二章第六节中相应的经典逻辑定义。模态表(modal tableau)和表证明是带标签的二叉树。现在这些标签(仍然称为表的表值(entries))，要么是标号力迫断言(即形如 $Tp \Vdash \varphi$ 或 $Fq \Vdash \varphi$ 的标记，其中 φ 为给定语言的语句)，要么是可达性断言 pSq。我们把 $Tp \Vdash \varphi$ 读作 p 力迫 φ，并把 $Fp \Vdash \varphi$ 读作 p 不力迫 φ。

因为在每个可能世界中我们所使用的是常规的谓词逻辑，所以命题联结词$\vee$，$\wedge$，$\neg$和$\rightarrow$的原子表，除了它们的表值现在是标号力迫断言之外，与第一章第四节或第二章第六节中的经典定义是一样的。量词$\forall$和$\exists$原子表的设计，既考虑到原先谓词逻辑所关心的问题，也反映了可能世界的论域在可达关系下的单调性假设。因此仍然要求只有“新”常元才能用于见证一个真存在语句或者用作假全称语句的反例。粗略地说，“新”常元就是前面没有做过任何承诺的常元，例如，不在$\mathcal{L}$中或在表中尚未出现的任何常元。另一方面，考虑一个真全称语句 $Tp \Vdash \forall x\varphi(x)$。在经典谓词逻辑中，我们可以用任何常元替换被全称量词所约束的变元 x。这里我们只能对$C(p)$或某个满足 qSp 的 $C(q)$中的那些常元 c 得出结论 $Tp \Vdash \varphi(c)$。这个思想可以翻译为，要求 c 在$\mathcal{L}$中或者已出现在当前路径的某个力迫断言中，且该断言包含 p 或者某个使得 qSp 出现在当前路径上的 q。这里的要点是，如果 qSp 并且 c 在 $C(q)$中，那么根据单调性 c 一定也在 $C(p)$中。在模态表的定义中，我们把这些常元称为“任何适当的 c”。当然，“新的”与“适当的”常元的形式定义也将在表定义中给出。

定义中另外一个关键要素是对于以$\Box$或$\Diamond$开头的标号力迫语句的处理。在经典逻辑中，对于通过表的展开而建立的结构，其元素是那些出现在该表的某条路径上的常元符号。我们现在要建立的是一个完整的框架。出现在表某条路径 P 的表值中的那些 p 和 q 构成了该框架的可能世界。我们还必须沿着表的每一条路径确定相应的可达关系 S。把这一信息直接包含在路径上，便于构造框架。因此我们允许在表中的表值出现形如 pSq 的事实，其中 p 和 q 出现在路径上该表值之前的某些标号力迫断言中。在$\Box$和$\Diamond$的原子表中含有这样的表值。例如，从 $Tp \Vdash \Diamond\varphi$ 可(语义地)得出结论：存在某 q，使得 pSq 且 $Tq \Vdash \varphi$。因此，$Tp \Vdash \Diamond\varphi$ 的原子表把 pSq 和 $Tq \Vdash \varphi$ 都放在其路径上，其中 q 为某个新世界(即尚未出现在当前所展开的表中)。另一方面，$Tp \Vdash \Box\varphi$ 的原子表反映了这样的思想：$p \Vdash \Box\varphi$ 的意思是 φ 在每个使得 pSq 的世界 q 中为真。对于任何适当的 q，即那些我们已知 pSq 成立的 q，亦即当前路径上 pSq 中的 q，该原子表把断言 $Tq \Vdash \varphi$ 放在此路径上。这样，我们沿着表的每一条路径建立了一个相应的框架。

我们现在形式地定义各个原子表。

定义 3.1(原子表) 我们从某给定模态语言$\mathcal{L}$开始。令$\mathcal{L}_C$为在$\mathcal{L}$中添加新常元 c_i($i \in \mathcal{N}$)

所得的扩张。图 43 列出了(语言$\mathcal{L}$的)原子表。在所列表中,φ 和 ψ 如果不带量词,那么它们是语言$\mathcal{L}_C$中任何语句。如果带量词,那么它们是以 x 为唯一变元的公式。

TA*t* $Tp \Vdash \varphi$ 对于任何原子语句 φ 和任何 p		FA*t* $Fp \Vdash \varphi$ 对于任何原子语句 φ 和任何 p	
T∨ $Tp \Vdash \varphi \vee \psi$ $Tp \Vdash \varphi$　$Tp \Vdash \psi$	F∨ $Fp \Vdash \varphi \vee \psi$ $Fp \Vdash \varphi$ $Fp \Vdash \psi$	T∧ $Tp \Vdash \varphi \wedge \psi$ $Tp \Vdash \varphi$ $Tp \Vdash \psi$	F∧ $Fp \Vdash \varphi \wedge \psi$ $Fp \Vdash \varphi$　$Fp \Vdash \psi$
T→ $Tp \Vdash \varphi \to \psi$ $Fp \Vdash \varphi$　$Tp \Vdash \psi$	F→ $Fp \Vdash \varphi \to \psi$ $Tp \Vdash \varphi$ $Fp \Vdash \psi$	T¬ $Tp \Vdash \neg\varphi$ $Fp \Vdash \varphi$	F¬ $Fp \Vdash \neg\varphi$ $Tp \Vdash \varphi$
T∃ $Tp \Vdash (\exists x)\varphi(x)$ $Tp \Vdash \varphi(c)$ 对于某个新的 c	F∃ $Fp \Vdash (\exists x)\varphi(x)$ $Fp \Vdash \varphi(c)$ 对于任何适当的 c	T∀ $Tp \Vdash (\forall x)\varphi(x)$ $Tp \Vdash \varphi(c)$ 对于任何适当的 c	F∀ $Fp \Vdash (\forall x)\varphi(x)$ $Fp \Vdash \varphi(c)$ 对于某个新的 c
T□ $Tp \Vdash \Box\varphi$ $Tq \Vdash \varphi$ 对于任何适当的 q	F□ $Fp \Vdash \Box\varphi$ pSq $Fq \Vdash \varphi$ 对于某个新的 q	T◇ $Tp \Vdash \Diamond\varphi$ pSq $Tq \Vdash \varphi$ 对于某个新的 q	F◇ $Fp \Vdash \Diamond\varphi$ $Fq \Vdash \varphi$ 对于任何适当的 q

图 43

提醒 在(T□)和(F◇)中我们允许不存在适当的 q 这种可能,这时(T□)和(F◇)的实例分别为 $Tp \Vdash \Box\varphi$ 和 $Fp \Vdash \Diamond\varphi$。

这个表的形式定义很像第二章第三节中经典逻辑的表定义。

定义 3.2 我们继续使用给定的模态语言$\mathcal{L}$及其用常元进行的扩张$\mathcal{L}_C$。我们再固定一个集合$\{p_i \mid i \in \mathcal{N}\}$,其中元素可用作力迫断言中的 p 和 q。($\mathcal{L}$的)**模态表**是一个用力迫断言或者可达关系作为节点标记的二叉树;两种标记都称为该表的**表值**。($\mathcal{L}$的)模态表的归纳定义如下。

i) 每个原子表 τ 是一个表。在情形(T∃)和(F∀)中要求"c 是新的",其意思是,c 是添加到$\mathcal{L}$得到$\mathcal{L}_C$的那些常元 c_i 之一,且 c 不在 φ 中出现。在(F∃)和(T∀)中,短语"任何适当

的 c"的意思是指 $\mathcal{L}$ 或 φ 中任何常元。在(F□)和(T◇)中要求 q 是新的，是指 q 是任何不同于 p 的 p_i。在本情形中，由于没有适当的 q，所以(T□)和(F◇)中短语"任何适当的 q"就意味着该表是 $Tp \Vdash \Box\varphi$ 或 $Fp \Vdash \Diamond\varphi$。

ii)如果 τ 是有穷表，P 是 τ 上的一条路径，E 是 τ 出现在 P 上的表值，且 τ' 是在 τ 中将根表值为 E 的原子表连接到 P 末端所得的树，那么 τ' 也是一个表。

这里，情形(T∃)和(F∀)中要求"c 是新的"，是指 c 为 c_i 之一(从而不在 $\mathcal{L}$ 中)且 c 不出现在 τ 的表值中。在(F∃)和(T∀)中，短语"任何适当的 c"的意思是指，c 是 $\mathcal{L}$ 中任何常元，或者是出现在 P 上任何形如 $Tp \Vdash \psi$，$Fp \Vdash \psi$，$Tq \Vdash \psi$ 或 $Fq \Vdash \psi$ 的表值，使得 qSp 也出现在 P 上。

在(F□)和(T◇)中要求"q 是新的"，是指选取一个不出现在 τ 上的 p_i 作为 q。在(T□)和(F◇)中短语"任何适当的 q"意思是指，我们可以选取 P 上任何 pSq 中的 q。

iii)如果 τ_0，τ_1，…，τ_n，…，是有穷表的一个序列，使得对任何 $n \geq 0$，τ_{n+1} 由 τ_n 应用ii)得来，那么 $\tau = \bigcup \tau_n$ 也是一个表。

同经典表的定义一样，我们要求在条款ii)中连接一个原子表到 P 的末端时，其根表值 E 仍要写出来，以保证最后得到的表是完成的。需要这样做的原子表有(F∃)，(T∀)，(T□)和(F◇)。但是为了符号上的简便，在我们的例子中一般不再重复写出根表值。下面的表证明的定义方式是我们所熟悉的。

定义 3.3(表证明)　令 τ 为一模态表而 P 是 τ 的一条路径。

i)P 是**矛盾的**，如果对于某个力迫断言 $p \Vdash \varphi$，$Tp \Vdash \varphi$ 与 $Fp \Vdash \varphi$ 都作为表值出现在 P 上。

ii)τ 是**矛盾的**，如果 τ 的每条路径是矛盾的。

iii)τ 是 φ 的**证明**，如果 τ 是一个根标记为 $Fp \Vdash \varphi$ 的有穷矛盾模态表。φ 是**可证明的**，记为 $\vdash\varphi$，如果存在 φ 的一个证明。

注意，同经典逻辑一样，如果存在一个根节点为 $Fp \Vdash \varphi$ 的矛盾表，那么就存在一个有穷的矛盾表，即 φ 的一个证明：当一条路径变矛盾时就不再继续展开。由于每条路径是有穷的，根据库尼西引理，整个树是有穷的。因此，定义中要求证明是有穷的(表)不影响任何语句证明的存在性。还有一个观点是，我们可以要求在表定义的ii)中的路径 P 为不矛盾的，这也不会影响证明的存在。因此，在实践中构造证明时我们用⊗标记矛盾路径，并停止沿着该路径继续展开表。

在处理模态逻辑表法的可靠性与完全性之前，先看一些模态证明的例子。注意，它们是简略的表，我们一般不再重复写出被展开项。我们在表的左边用数字标出其各个分层，在其右边指出每个分层是由上面哪一层的原子表产生的。

例 3.4　对于不含模态算子的语句，在经典谓词逻辑与模态逻辑的表之间存在一个自然对应。把标号力迫断言 $Tp \Vdash \varphi$ 和 $Fp \Vdash \varphi$ 分别替换为对应的标号语句 $T\varphi$ 和 $F\varphi$，可将模态表对应到经典表。(另一个方向的对应必须考虑第二章定义6.1中新常元的不同含义。)注意到，该对应关系把证明对应到证明(见习题1)。

例 3.5　$\varphi \to \Box\varphi$，有时称为必然模式(scheme of necessitation)，不是永真的。图44尝试给出其证明。

1　$Fw \Vdash \varphi \to \Box\varphi$

2　$Tw \Vdash \varphi$　由1

3　$Fw \Vdash \Box\varphi$　由1

4　wSv 对某个新的 v　由3

5　$Fv \Vdash \varphi$　由3

图　44

这个失败的尝试隐含着一个框架反例 $\mathcal{C}$，其中 W =

$\{w, v\}$，$S=\{(w, v)\}$且结构使得 φ 在 w 中成立但在 v 中不成立。这样的框架表明 $\varphi\to\Box\varphi$ 不是永真的，因为在这个框架中，w 不力迫$\varphi\to\Box\varphi$。

例 3.6 类似地，$\Box\varphi\to\varphi$ 不是永真的，这可从图 45 中所尝试的证明看出。

1	$Fw\Vdash\Box\varphi\to\varphi$	
2	$Tw\Vdash\Box\varphi$	由1
3	$Fw\Vdash\varphi$	由1

图 45

这里所隐含的框架反例，包含一个世界 $W=\{w\}$ 与一个空的可达关系 S，且使 φ 在 w 上为假。这表明$\Box\varphi\to\varphi$ 不是永真的。

□的不同解释诱使人们认为$\Box\varphi\to\varphi$ 应当是永真的。例如，大概所有哲学家都会同意，如果 φ 必然为真，那么它事实上为真。另一方面，大多数但也许不是所有的认知学家会认为，如果我知道 φ，那么它也一定为真。最后，很少有人会声称，(对于任何 φ)如果我相信 φ，那么 φ 就为真。习惯上，把$\Box\varphi\to\varphi$ 称为"T"或"知识公理"。在□的许多解释中，它应当是永真的。从上面的尝试性证明可以看出，假如我们知道 wSw 的话，那么很快可以得到所需要的矛盾。因此，在 T 与可达关系的自反性之间存在某种联系。事实上，不仅 T 在所有带自反可达关系的框架中是永真的，而且反过来任何在所有这样框架中永真的语句都能由 T 推导出来。在第五节中精确地说明了这个以及其他像这样的对应关系。

例 3.7 我们在图 46 中证明$\Box(\forall x)\varphi(x)\to(\forall x)\Box\varphi(x)$是可证明的。

注意，第 6 行和第 8 行的推导使用了单调性，这与例 2.5 中的语义论证是相对应的。

例 3.8 图 47 给出$\Box(\varphi\to\psi)\to(\Box\varphi\to\Box\psi)$的一个表证明。

1	$Fw\Vdash\Box(\forall x)\varphi(x)\to(\forall x)\Box\varphi(x)$	
2	$Tw\Vdash\Box(\forall x)\varphi(x)$	由1
3	$Fw\Vdash(\forall x)\Box\varphi(x)$	由1
4	$Fw\Vdash\Box\varphi(c)$	由3
5	wSv	由4
6	$Fv\Vdash\varphi(c)$	由4
7	$Tv\Vdash(\forall x)\varphi(x)$	由2，5
8	$Tv\Vdash\varphi(c)$	由7
	$\otimes$	由6，8

图 46

1	$Fw\Vdash\Box(\varphi\to\psi)\to(\Box\varphi\to\Box\psi)$			
2	$Tw\Vdash\Box(\varphi\to\psi)$			由1
3	$Fw\Vdash\Box\varphi\to\Box\psi$			由1
4	$Tw\Vdash\Box\varphi$			由3
5	$Fw\Vdash\Box\psi$			由3
6	wSv		新的v	由5
7	$Fv\Vdash\psi$			由5
8	$Tv\Vdash\varphi$			由4，6
9	$Tv\Vdash\varphi\to\psi$			由2，6
10	$Fv\Vdash\varphi$	$Tv\Vdash\psi$		由9
11	$\otimes$	$\otimes$		由8，7

图 47

(图 47 所论证的语句在第六节希尔伯特式的模态逻辑证明系统中被用作公理模式。)

例 3.9　图 48 给出了 $\forall x\Box\varphi(x)\rightarrow\Box\forall x\varphi(x)$ 的一个错误证明。

第 7 行的推导是错误的。根据第 6 行我们可以把 c 用于有关 v 或者任何 v 可达世界的力迫断言中，但我们没有理由把它用于关于 w 的断言中。如同例 2.7 中那样，这一步推导在常论域框架的分析中是可以的。(见第四节习题 8。)

例 3.10　$(\forall x)\neg\Box\varphi\rightarrow\neg\Box(\exists x)\varphi$ 不是永真的。图 49 中以此公式开头的表不是一个证明。但是它提醒了我们如何构造一个框架反例。令该框架有一个常论域 $C=\{c,\ d\}$；两个世界 w 和 v，使得从 w 可达 v；没有原子语句在 w 上为真，且语句 $\varphi(d)$ 在 v 上为真。这个框架提供了所需要的反例。

1	$Fw\Vdash(\forall x)\Box\varphi(x)\rightarrow\Box(\forall x)\varphi(x)$		
2	$Tw\Vdash(\forall x)\Box\varphi(x)$		由1
3	$Fw\Vdash\Box(\forall x)\varphi(x)$		由1
4	wSv		由3
5	$Fv\Vdash(\forall x)\varphi(x)$		由3
6	$Fv\Vdash\varphi(c)$	新的c	由5
7	$Tw\Vdash\Box\varphi(c)$		由2
8	$Tv\Vdash\varphi(c)$		由7
	$\otimes$		

图　48

1	$Fw\Vdash(\forall x)\neg\Box\varphi\rightarrow\neg\Box(\exists x)\varphi$		
2	$Tw\Vdash(\forall x)\neg\Box\varphi$		由1
3	$Fw\Vdash\neg\Box(\exists x)\varphi$		由1
4	$Tw\Vdash\Box(\exists x)\varphi$		由3
5	$Tw\Vdash\neg\Box\varphi(c)$		由2
6	$Fw\Vdash\Box\varphi(c)$		由5
7	wSv		由6
8	$Fv\Vdash\varphi(c)$		由6
9	$Tv\Vdash(\exists x)\varphi$		由4,7
10	$Tv\Vdash\varphi(d)$	新的d	由9

图　49

同定义逻辑后承的语义时一样，我们必须谨慎地定义从语句集 Σ(其中语句常称为前件)出发的模态表证明。这个定义应该符合这样的直觉：我们只需考虑所有前件都在其中被力迫的那些框架。为此，我们允许对于任何适当的可能世界 p 与任何 $\varphi\in\Sigma$，在表中加入形如 $Tp\Vdash\varphi$ 的表值。

定义 3.11　令 Σ 为某模态语言的语句集合，其中的语句称为前件。**从 Σ 出发的模态表**(modal tableau from Σ)定义与定义 3.2 中简单模态表是一样的，除了一个附加的构成规则：

(ii′)如果 τ 是一个从 Σ 出发的有穷表，$\varphi\in\Sigma$，P 是 τ 的一条路径，且 p 是在 P 上某标号力迫断言中出现的可能世界，那么把 $Tp\Vdash\varphi$ 连接到 P 的末端就构成了从 Σ 出发的另一个表 τ'。

现在那些用来定义表证明的概念可以从定义 3.3 移植到从 Σ 出发的表证明上，只要把其中的“表”替换为“从 Σ 出发的表”即可。我们用 $\Sigma \vdash \varphi$ 表示 φ 是从 Σ 出发可证明的，即存在 φ 的一个从 Σ 出发的证明。

例 3.12 图 50 给出了 $\Box\forall x\varphi(x)$ 的一个从前件 $\forall x\varphi(x)$ 出发的表证明。

1	$Fp \Vdash \Box(\forall x)\varphi(x)$		
2	pSq		由1
3	$Fq \Vdash (\forall x)\varphi(x)$		由1
4	$Fq \Vdash \varphi(c)$	新的c	由3
5	$Tq \Vdash (\forall x)\varphi(x)$		前件
6	$Tq \Vdash \varphi(c)$		由5
	$\otimes$		

图 50

习题

1. 把例 3.4 所描述的对应关系精确化，并证明它把经典谓词逻辑的表证明变换为模态逻辑的表证明。反过来，如果 τ 是经典逻辑语句 φ 的模态表证明，描述另一个方向的适当的变换，并证明它把 τ 变换为 φ 的经典证明。

在习题 2～8 中，令 φ 和 ψ 为不带自由变元或在适当的情形中仅含自由变元 x 的任何公式。给出它们的模态表证明。

2. $\neg\Diamond\neg\varphi \to \Box\varphi$
3. $\forall x\Box\varphi(x) \to \exists x\Box\varphi(x)$
4. $\Box\Box((\varphi \vee (\forall x)\psi(x)) \to (\forall x)(\varphi \vee \psi(x)))$，$x$ 在 φ 中不自由
5. $\neg\Diamond(\neg(\varphi \wedge (\exists x)\psi(x)) \wedge (\exists x)(\varphi \wedge \psi(x)))$，$x$ 在 φ 中不自由
6. $\Box(\varphi \vee \neg\psi) \to (\Diamond\psi \to \Diamond\varphi)$
7. $\Box(\exists x)(\varphi \wedge \psi(x)) \to \Box(\varphi \wedge (\exists x)\psi(x))$，$x$ 在 φ 中不自由
8. $\Diamond(\exists x)(\varphi(x) \to \Box\psi) \to \Diamond((\forall x)\varphi(x) \to \Box\psi)$，$x$ 在 ψ 中不自由

给出下面公式的模态表证明：

9. $\varphi \Vdash \Box\varphi$
10. $(\varphi \to \Box\varphi) \Vdash \Box\varphi \to \Box\Box\varphi$
11. $\forall x\varphi(x) \Vdash \forall x\Box\varphi(x)$
12. **常元定理**：令 $\varphi(x_1, \cdots, x_n)$ 为模态语言$\mathcal{L}$的公式，其所有自由变元都已显示出来。再令 $c_1, \cdots, c_n$ 为不在$\mathcal{L}$中的常元符号。证明：$\forall x_1 \cdots \forall x_n\varphi(x_1, \cdots, x_n)$ 是表可证的，当且仅当 $\varphi(c_1, \cdots, c_n)$ 是表可证的。从语法上论证，给定其中一个公式的证明可以构造出另外一个公式的证明。

第四节　可靠性和完全性

本节第一个目标是证明：(与在经典逻辑中一样)模态逻辑中可证明性蕴涵永真性。类似于经典可靠性定理的证明(第二章定理 7.2)，先证明，一个框架如果与某表的根节点“一致”，那么它也与该表某条路径上的每一个表值“一致”。在经典情形中(第二章定义 7.1)，我们在表中构造出这样的路径并定义一个结构，其论域由出现在该路径标号语句中的常元 c 构成。在模态情形中，我们应当定义一个可能世界集 W，并且对于每个 $p \in W$，根据出现在该路径上的常元定义一个结构。W 由在该路径的力迫断言中出现的 p 组成。然后根据该路径上出现的断言 pSq 定义 W 上的可达关系。

定义 4.1 设$\mathcal{C} = (V, T, \mathcal{C}(p))$是模态语言$\mathcal{L}$的框架，$\tau$ 是一个表，其根标记是$\mathcal{L}$中语句 φ 的力迫断言，且 P 是 τ 的一个条路径。令 W 为在 P 的力迫断言中出现的 p 的集合，并令 S 为 W 上的可达关系，它由 P 上的那些断言 pSq 所确定。我们说$\mathcal{C}$与 P **一致**，如果存在映射 f 和 g 使得

(i) f 是从 W 到 V 上的保持可达关系的映射，即 $pSq \Rightarrow f(p)Tf(q)$。(注意，$f$ 不一定是一

一映射。)

(ii) g 把 P 上任何力迫断言 $Tp \Vdash \psi$ 或 $Fp \Vdash \psi$ 的语句 ψ 中出现的常元 c 对应到 $\mathcal{L}(f(p))$ 中某个常元。此外，g 在 $\mathcal{L}$ 的常元上是恒等映射。我们以下面方式把 g 扩张为公式上的映射：把 ψ 中每个常元 c 替换为 $g(c)$ 得到 $g(\psi)$。

(iii) 如果 $Tp \Vdash \psi$ 在 P 上，那么 $f(p)$ 在 $\mathcal{C}$ 中力迫 $g(\psi)$，且如果 $Fp \Vdash \psi$ 在 P 上，那么 $f(p)$ 在 $\mathcal{C}$ 中不力迫 $g(\psi)$。

定理 4.2 设 $\mathcal{C}=(V, T, \mathcal{C}(p))$ 是模态语言 $\mathcal{L}$ 的框架，且 τ 是一个表，其根标记是关于 $\mathcal{L}$ 的语句 φ 的一个力迫断言。如果 $q \in V$ 且

(i) $Fr \Vdash \varphi$ 是 τ 的根且 q 在 $\mathcal{C}$ 中不力迫 φ

或者

(ii) $Tr \Vdash \varphi$ 是 τ 的根且 q 在 $\mathcal{C}$ 中力迫 φ，

那么在 τ 中存在一条与 $\mathcal{C}$ 一致的路径 P，其(如定义 4.1 所要求的)见证函数 f 把 r 映到 q。

根据表 τ 的构造对该定理进行归纳证明。在给出其细节之前，我们把该结果重新表述为标准版本的可靠性定理。

定理 4.3(可靠性，$\vdash\varphi \Rightarrow \vDash\varphi$) 如果存在(模态逻辑的)语句 φ 的一个(模态)表证明，那么 φ 是(模态)永真的。

证明(可靠性) φ 的模态表证明是根形如 $Fr \Vdash \varphi$ 的表 τ，其每条路径都是矛盾的。如果 φ 不是永真的，那么存在一个框架 $\mathcal{C}=(V, T, \mathcal{C}(p))$ 和一个 $q \in V$，使得 q 在 $\mathcal{C}$ 中不力迫 φ。现在应用定理 4.2 得到 τ 的一条路径 P 与函数 f 和 g 满足定义 4.1 中所列的性质。因为 τ 是矛盾的，存在一个 p 与一个语句 ψ，使得 $Tp \Vdash \psi$ 与 $Fp \Vdash \psi$ 都在 P 上。由定义 4.1(iii) 立刻得出矛盾。 □

我们把定理 4.2 的证明分成几个部分。首先，对应于定义 3.2 的 i) 和 18 个原子表，有 18 种基本情形。

引理 4.4 对于每个满足定理 4.2 的假设的原子表 τ，存在该定理结论中所要求的 P, f 和 g。

其次，对应于定义 3.2 中的 ii) 所选择的原子表类型，共有 16 种归纳情形。为了便于归纳，我们实际证明的是一个比该定理更强的结论。

引理 4.5 如果 f 和 g 是表 τ 的路径 P 与 $\mathcal{C}$ 一致的证据，且 τ' 由 τ 应用定义 3.2 的 ii) 得到，那么分别存在 P, f 和 g 的扩张 P', f' 和 g'，使得 f' 和 g' 是表 τ' 的路径 P' 与 $\mathcal{C}$ 一致的证据。

定理 4.2 是这两个引理的简单推论，所以在证明两个引理之前先给出该定理的证明。

证明(定理 4.2) 根据引理 4.4，定理对于原子表成立。然后根据引理 4.5，通过归纳法可以证明定理对于所有有穷表成立。其实，这也证明了定理对于无穷表成立：设 $\tau=\cup\tau_n$ 是定义 3.2 的 iii) 中所定义的无穷表。首先对 τ_0 应用引理 4.4 得到适当的 P_0, f_0 和 g_0。然后再依次对每个 τ_n 应用引理 4.5 构造 P_n, f_n 和 g_n。对于 τ 所要求的 P, f 和 g 分别是 P_n, f_n 和 g_n 的并。 □

证明(引理 4.4) 首先定义 $f(p)=r$，g 为 $\mathcal{L}_C$ 中常元上的恒等映射。根据定理的假设，这样定义的 f 和 g 表明原子表的根与 $\mathcal{C}$ 一致。这就完成了对于 T*At* 与 F*At* 的证明。对于其他

原子表的论证完全与引理 4.5 的对应情形相同。对于这些其他原子表进行归纳论证可以提供所需要的扩张。(也许从技术上来说，退化情形(T□)和(F◇)是例外，但在这些情形中结论与假设是相同的。)因此我们已经把该引理的证明归约到引理 4.5 的证明。 □

证明(引理 4.5) 首先注意到，如果 τ'不是通过在 P 的末端扩展 τ 得到的，那么 τ 的证据对于 τ'也起作用。因此，可设 τ'是由 τ 通过在 P 的末端连接一个原子表得到的。现在考虑定义 3.1 中 16 种原子表所对应的情形。

(T∃)，(F∀)，(F□)与(T◇)之外的其他情形不要求扩张 f 或 g。在这些情形中，根据归纳假设与力迫定义(定义 2.2)的相应情形(对于情形(F∃)和(T∀)，论域 $C(p)$上的单调性假设)，显然可知 τ'上由 P 扩展所得的各路径中必有一条满足引理的要求。

我们详细地论证情形(T∃)和(T◇)。情形(F∀)和(F□)是类似的，留作习题 1。在情形(T∃)中，P 上被展开的表值是 $Tp \Vdash (\exists x)\varphi(x)$。所得 P 的扩张 P'是唯一的，它是通过在 P 的末端加上 $Tp \Vdash \varphi(c)$得到的。根据我们的归纳假设，$f(p) \Vdash_{\mathcal{C}} g((\exists x)\varphi(x))$。根据存在语句的力迫定义(定义 2.2(v))，存在 $c' \in \mathcal{L}(p)$，使得 $f(p) \Vdash g(\varphi(c'))$。固定这个 c'并通过令 $g'(c) = c'$，把 g 扩张为 g'。现在显然，P'，$f' = f$ 和 g'满足引理的要求，即 f'和 g'见证 P'与$\mathcal{C}$一致。

最后，在情形(T◇)中，P 上被展开的表值是 $Tp \Vdash \Diamond\varphi$。所得 P 的扩张 P'是唯一的，它是通过在 P 的末端加上 pSq 和 $Tq \Vdash \varphi$ 得到的。根据我们的归纳假设，$f(p) \Vdash_{\mathcal{C}} g(\Diamond\varphi)$。因为 $g(\Diamond\varphi) = \Diamond g(\varphi)$，根据◇的力迫定义(定义 2.2(x))，存在 $q' \in V$，使得 $f(p)Tq'$且 $q' \Vdash g(\varphi)$。固定这个 q'并通过令 $f'(q) = q'$，把 f 扩张为 f'。现在显然，P'，f'和 $g' = g$ 满足引理的要求，即 f'和 g'见证 P'与$\mathcal{C}$一致。 □

我们下一个目标是证明表证明法对模态逻辑来说是完全的。首先定义(像第二章定义 6.9 那样的)一种方法，用来构造适当的以给定的标号力迫断言作为根标记的完全系统表。然后证明，对于该表上的任一条不矛盾路径 P，我们能建立一个与 P 一致的框架$\mathcal{C}$。因此，如果这个系统性方法应用于某个力迫断言 $Fp \Vdash \varphi$ 时不能产生 φ 模态表证明的话，那么我们将可以建立一个不力迫 φ 的框架，从而证明 φ 不是永真的。

首先把已约化的表值与完成表等概念(第二章定义 6.7)推广到模态逻辑。

注意，c_1，…，c_n，…是扩张语言$\mathcal{L}_C$的所有常元序列，而 p_1，p_2，…是备用的可能世界序列。为了方便，我们设 c_1 在$\mathcal{L}$中。

定义 4.6 令 $\tau = \cup\tau_n$ 是一个表，P 是 τ 的路径，E 是 P 上的表值，且 w 是 E 在 P 上的第 i 次出现(即 P 上第 i 个标记为 E 的节点)。

(i)w 在 P 上是**已约化的**(reduced)，如果下列情形之一成立：

(1)E 与(定义 3.1 中的)原子表(F∃)，(T∀)，(T□)或(F◇)的根在形式上不相同，且对于某个 j，τ_{j+1}是在 τ_j 中通过对 E 和 τ_j 中作为 P 的初始段的路径应用定义 3.2 的规则(ii)所得的表。(这时我们说 E 作为原子表的根表值出现在 P 上。)

(2)E 形如 $Fp \Vdash (\exists x)\varphi(x)$ 或 $Tp \Vdash (\forall x)\varphi(x)$(分别为情形(F∃)和(T∀))；在 P 上存在 E 的第 $i+1$ 个出现且

(a)当 qSp 出现在 P 上时，c_i 不出现在 P 的任何关于可能世界 q 的断言中，或者

(b)$Fp \Vdash \varphi(c_i)$或 $Tp \Vdash \varphi(c_i)$是 P 的表值。

(3)E 形如 $Tp \Vdash \Box\varphi$ 或 $Fp \Vdash \Diamond\varphi$(分别为情形(T□)和(F◇))；在 P 上存在 E 的第 $i+1$

个出现且

(a) pSp_i 不是 P 的表值，或者

(b) $Tp_i \Vdash \varphi$ 或 $Fp_i \Vdash \varphi$ 是 P 的表值。

(4) E 形如 pSq。

(ii) τ 是**完成的**，如果 τ 上每个表值的每次出现在含有它的每一条不矛盾路径上都是已约化的。否则它是**未完成的**。

如同在处理经典表时那样，一个形如 $Tp \Vdash (\forall x)\varphi(x)$ 的标号力迫断言，必须用我们语言中每个常元 c_i 实例化之后，才能说我们已经把它完成。另外，如果这里 p 力迫 $\Box\varphi$，那么 φ 必须被 p 的所有后继 q 所力迫。现在可以通过构造适当的完全系统表来证明：存在以任何给定的标号力迫断言为根的完成表。我们采用的方案与基于层字典序 $\leqslant_{LL}$（第二章定义 6.8）的经典情形相同。

引理 4.7 设 w 是 E 在表 τ 的路径 P 上的第 i 次出现且在 P 上是已约化的。如果 τ' 是 τ 的一个扩张，而 P' 是 P 在 τ' 中的一个扩张，那么 w 在 P' 上不是已约化的唯一原因是

(i) E 形如 $Fp \Vdash \exists x\varphi(x)$ ($Tp \Vdash \forall x\varphi(x)$)，且当 qSp 出现在 P 上时，c_i 不出现在 P 的任何关于可能世界 q 的断言中，但 c_i 出现在 P' 上这样的断言中，且 $Fp \Vdash \varphi(c_i)$ ($Tp \Vdash \varphi(c_i)$) 不是 P' 的表值；或者

(ii) E 形如 $Tp \Vdash \Box\varphi$ ($Fp \Vdash \Diamond\varphi$)，且 pSp_i 出现在 P' 上但不出现在 P 上，且 $Fp_i \Vdash \varphi$ ($Tp_i \Vdash \varphi$) 不出现在 P' 上。

证明 根据定义该结论是显然的。 □

定义 4.8 我们用归纳法定义**由语句 φ 开始的完全系统模态表**（complete systematic modal tableau, CSMT）如下。

(i) τ_0 是根为 $Fp_1 \Vdash \varphi$ 的原子表。根据下面的规定，这个原子表是唯一确定的。在情形 (F∃) 和 (T∀) 中，我们用常元 c_1。在情形 (T∃) 和 (F∀) 中，我们用下标 i 最小的 c_i。而在情形 (F□) 和 (T◇) 中，我们用不在根部出现的最小的 p_i。（注意，情形 (T□) 和 (F◇) 的表只有根表值。它是完成表并且构成 CSMT。）

应用归纳法，设在第 n 步得到表 τ_n。如果 τ_n 是完成的，那么我们停止构造。否则，令 w 为 τ_n 中层字典序最小的节点且 w 含有表值 E 的一次出现，其中 E 是在 τ_n 某条不矛盾路径 P 上未已约化的表值。我们现在应用下面的做法之一把 τ_n 扩展为 τ_{n+1}：

(ii) 如果 E 不是形如情形 (F∃)，(T∀)，(T□) 或 (F◇) 的根节点，那么把以 E 为根的原子表接到 τ 中含 w 的所有不矛盾路径的末端。对于类型为 (T∃) 或 (F∀) 的 E，我们用尚未在表中出现的最小的常元 c_j。如果 E 的类型是 (F□) 或 (T◇)，那么选取 p_j 作为 q，其中 j 是使得 p_j 尚未出现在表中的最小下标。

(iii) 如果 E 的类型是 (F∃) 或 (T∀)，且 w 是 E 在 P 上的第 i 次出现，那么我们把相应的原子表接到 P 的末端，其中 c 使用下标 j 最小的 c_j，满足 c_j 是适当的且 $Fp \Vdash \varphi(c_j)$ 或 $Tp \Vdash \varphi(c_j)$ 作为一个表值不出现在 P 上。如果没有这样的 c_j，c 就取 c_1。

(iv) 如果 E 的类型是 (T□) 或 (F◇)，且 w 是 E 在 P 上的第 i 次出现，那么我们把相应的原子表接到 P 的末端，其中 q 使用下标 j 最小的 q_j，满足 q_j 是适当的且 $Tq_j \Vdash \varphi$ 或 $Fq_j \Vdash \varphi$ 作为一个表值不出现在 P 上。如果对于每个适当的 q_j，$Tq_j \Vdash \varphi$（或 $Fq_j \Vdash \varphi$）已经出现在 P 上，那么我们就重复 $Tq_j \Vdash \varphi$（或 $Fq_j \Vdash \varphi$），其中 j 是最小的下标，使得 q_j 是适当的。（根据

假设 E 在 P 上不是已约化的，至少存在一个适当的 q_j。)

表序列 τ_n 的并 τ 就是由 φ 开始的 CSMT。

注意，一般来说一个 CSMT 是无穷的表(即使 S 是有穷的)。重点在于它总是一个完成表。

引理 4.9 如果 p_iSp_j 作为一个表值出现在某个 CSMT 上，那么 $i<j$。

证明 只有在情形(F□)和(T◇)中，根据定义 3.8(ii)把形如 p_iSp_j 的表值放在表上时，新的可能世界 p_j 和可达关系新的实例才会出现在一个 CSMT$\tau=\cup\tau_n$ 上。因此它们是按数字的次序放到该 CSMT 上去的，从而当 p_iSp_j 在该树上时就有 $i<j$。 □

命题 4.10 每个 CSMT 都是完成的。

证明 对于给定的 CSMTτ，考虑任何表值 E 及其在某条不矛盾路径 P 上的任何未约化的出现 w。(如果不存在这样的 w，那么根据定义，τ 是完成的。)设 E 所做的力迫断言是关于某可能世界 p_m 的，w 是 E 在 P 上的第 i 次出现，且在 τ 上有 n 个节点其层字典序小于 w。令 k 足够大使得

(i) E 的出现 w 在 τ_k 中。

(ii) 如果 p_mSp_i 在 P 上，那么它在 τ_k 中的 P 上。

(iii) 如果任何关于某个可能世界 p_j 的断言(该断言使得 p_jSp_m 出现在 P 上)和常元 c_i 出现在 P 上，那么 p_jSp_m 与某个涉及 p_j 和 c_i 的断言出现在 τ_k 中的 P 上。

注意到，根据引理 4.9 仅存在有穷多的 p_j 与(iii)相关，从而我们可以发现一个足够大的 k 适合上述所有条件。

显然，根据 CSMT 定义，在构造 τ_{k+n+1} 时，必须在 P 上约化 w。再根据引理 4.7，一旦这样被约化，w 就永远是已约化的。因此在 τ 的每条不矛盾路径上，每个表值的每个出现都是已约化的，这正是所需要的结果。 □

我们现在可以证明一个完全性定理，只要论证了由 $Fp_1\Vdash\varphi$ 开始的 CSMT 要么是一个证明，要么给我们提供了一个框架反例。

定理 4.11 设 $\tau=\cup\tau_n$ 是一个 CSMT，P 是 τ 中一条不矛盾路径。我们定义 P 对应的框架 $\mathcal{C}=(W, S, \mathcal{C}(p))$ 如下：

(i) W 是所有出现在 P 上力迫断言中的 p_i 的集合。S 是 P 上所有 p_iSp_j 所对应的二元组 (p_i, p_j) 构成的集合。

(ii) 对于任何 $p_i\in W$，通过对 i 归纳定义 $C(p_i)$ 为 $\mathcal{L}$ 的所有常元和出现在 P 上力迫断言 $Tq\Vdash\psi$ 或 $Fq\Vdash\psi$ 中的使得 qSp 的所有其他常元所构成的集合。(注意，根据引理 4.10，如果 p_jSp_i 出现在 P 上，那么 $j<i$。因此 $C(p_i)$ 的归纳定义合理。)

(iii) 对于每个 $p\in W$，$\mathcal{C}(p)$ 的定义是：每个原子语句 ψ 在 $\mathcal{C}(p)$ 为真，当且仅当 $Tp\Vdash\psi$ 出现在 P 上。(提醒：我们采用这样的惯例，即每个 $c\in C(p)$ 在 $\mathcal{L}(p)$ 中用它自己命名。)

如果令 f 和 g 分别为 W 和出现在 P 上的所有常元集合上的恒等函数，那么它们见证 $\mathcal{C}$ 与 P 一致。

证明 首先，根据定义 2.1，上述定义 $\mathcal{C}$ 的各条款保证了 $\mathcal{C}$ 是 $\mathcal{L}$ 的框架。注意，$\mathcal{L}(p)$ 中每个常元 c 以自己命名。

我们现在希望证明：(f 和 g 见证) P 与 $\mathcal{C}$ 一致。对出现在 P 上力迫断言中的语句 φ 的深度进行归纳。归纳的关键在于，根据命题 4.9，P 上每个表值的每个出现是已约化的。

(i)原子 φ：如果 $Tp \Vdash \varphi$ 出现在 P 上，那么 φ 在 $\mathcal{C}(p)$ 中为真，从而被 p 力迫。如果 $Fp \Vdash \varphi$ 出现在 P 上，那么因为 P 是不矛盾的，所以 $Tp \Vdash \varphi$ 不出现在 P 上。由于这是 p 力迫 φ 的唯一可能的方式，我们可得所要求的结论：p 不力迫 φ。

对于每一种归纳情形，我们根据定义 4.6 中对应的条款和力迫定义(定义 2.2)以及本定理的归纳假设，进行诸个处理。这里考虑一些有代表性的情形，其他的留作习题。

(ii)命题联结词：设 φ 用一个联结词构成，例如，φ 是$(\varphi_1 \vee \varphi_2)$。因为 τ 是完成的，所以如果 $Tp \Vdash \varphi$ 出现在 P 上，那么 $Tp \Vdash \varphi_1$ 或 $Tp \Vdash \varphi_2$ 出现在 P 上。根据归纳假设，假如 $Tp \Vdash \varphi_1$ 出现在 P 上，那么 p 力迫 φ_1，从而根据力迫的定义(定义 2.2 的(vii))，可得所要求的结果，即 p 力迫 φ。类似地，如果 $Fp \Vdash \varphi$ 出现在 P 上，那么 $Fp \Vdash \varphi_1$ 和 $Fp \Vdash \varphi_2$ 出现在 P 上。因此根据归纳和定义 2.2的(vii)，可得所要求的结果，即 p 不力迫 φ。其他经典的命题联结词可类似地处理。(见习题 2。)

(iii)量词：设 φ 形如$(\forall v)\psi(v)$。如果 w 是 $Tp \Vdash (\forall v)\psi(v)$ 在 P 上的第 i 次出现，那么存在 $Tp \Vdash (\forall v)\psi(v)$ 在 P 上的第 $i+1$ 次出现。此外，如果 $c_i \in C(p)$，那么 $Tp \Vdash \psi(c_i)$ 出现在 P 上。因此，如果 $Tp \Vdash (\forall v)\psi(v)$ 出现在 P 上，那么对每个 $c \in C(p)$，$Tp \Vdash \psi(c)$ 都出现在 P 上。因为$\psi(c)$的深度比$(\forall v)\psi(v)$低，所以对每个 $c \in C(p)$，p 力迫 $\psi(c)$。因此，根据定义 2.2 的(iv)，p 力迫$(\forall v)\psi(v)$。如果 $Fp \Vdash (\forall v)\psi(v)$ 出现在 P 上，那么再次因为 τ 是完成的，对于某个 c，$Fp \Vdash \psi(c)$ 出现在 P 上。根据归纳假设，p 不力迫 $\psi(c)$。因此，根据定义 2.2 的(iv)得所要求的结果：p 不力迫$(\forall v)\psi(v)$。

对于存在量词的分析是类似的，留作习题 3。

(iv)模态算子：如果 $Tp \Vdash \Box\varphi$ 和 pSq 出现在 P 上，那么因为 τ 是完成的，所以 $Tq \Vdash \varphi$ 出现在 P 上。(注意，表是完成的可保证：如果 $Tp \Vdash \Box\varphi$ 出现，那么它就有无穷多次出现，从而有某个第 j 次出现，其中 $q = p_j$。)因此根据归纳，q 力迫 φ，且根据定义 2.2 的(ix)，p 力迫$\Box\varphi$。

如果 $Fp \Vdash \Box\varphi$ 出现在 P 上，那么对于某个 q，pSq 和 $Fq \Vdash \varphi$ 都出现在 P 上。因此根据归纳，q 不力迫 φ，且根据定义，q 是 p 的一个后继。所以根据定义 2.2 的(ix)，p 不力迫$\Box\varphi$。

$\Diamond$情形是类似的，留作习题 4。

至此结合习题 2 完成本证明。 □

我们现在可以陈述标准形式的完全性定理。

定理 4.12(完全性，$\vDash\varphi \Rightarrow \vdash\varphi$) *如果模态逻辑语句 φ(在框架语义中)是永真的，那么它有一个(模态)表证明。*

证明 设 φ 是永真的。考虑由根 $Fp_1 \Vdash \varphi$ 开始的 CSMT τ。根据定义，τ 的每个矛盾路径是有穷的。因此，如果 τ 的每条路径是矛盾的，那么根据库尼西引理，τ 是有穷的。特别地，如果 τ 不是 φ 的表证明，那么就存在一条不矛盾路径 P。然后根据定理 4.9 可知，存在一个框架$\mathcal{C}$，其中 p_1 不力迫 φ。因此，φ 不是永真的，从而得所需要的矛盾。 □

现在可依照前面的做法，把可靠性和完全性定理推广到逻辑后承的模态概念和由前件出发的表演绎。

定理 4.13(可靠性，$\Sigma\vdash\varphi \Rightarrow \Sigma\vDash\varphi$) *如果模态逻辑语句 φ 有一个从语句集 Σ 出发的模态表证明，那么 φ 是 Σ 的逻辑后承。*

证明 可靠性定理(定理 4.3)的基本组成部分(定理 4.2)的证明表明，如果 τ 是一个从

Σ 出发的表，且框架$\mathcal{C}$力迫每个 $\psi\in\Sigma$ 并且与 τ 的根一致，那么$\mathcal{C}$与 τ 的某条路径 P 一致。唯一的新情况在于，表可以通过以下的方式进行扩展：对任何 $\psi\in\Sigma$，把断言 $Tp\Vdash\psi$ 添加到任何出现 p 的路径末端。可以很容易地通过改造引理 4.5 的证明来处理这种不同情况。因为(根据假设)对于$\mathcal{C}$的每个可能世界 q，有 $q\Vdash_{\mathcal{C}}\psi$，所以引理4.5 证明中的归纳假设在这个新情形中也可以立刻得到验证。现在从这个结果可以推导出本定理，正如从定理 4.2 推出定理 4.3那样。 □

要证明从前件出发的演绎的完全性定理，我们需要明确从前件集出发的 CSMT 这个概念。

定义 4.14(CSMT) 我们用归纳法定义一个由语句 φ 开始的**从语句集 Σ 出发的完全系统模态表**如下。τ_0 是根为 $Fp_1\Vdash\varphi$ 的原子表。作为归纳步，我们像调整第二章定义 6.9 中的 CST 那样调整定义 4.8 以容纳推理前件。令 $\Sigma=\{\psi_j\mid j\in\mathcal{N}\}$。在构造的偶数阶段我们按照定义 4.8 的(i) ~ (iv)进行。在奇数阶段 $n=2k+1$，对于每个 i，$j<k$，我们把 $Tp_i\Vdash\psi_j$ 接到 τ_n 中有 p_i 出现的每条不矛盾路径 P 上。我们一直不停地构造，直到对于每个 $\psi\in\Sigma$，$Tp\Vdash\psi$ 出现在每条含有 p 的不矛盾路径 P 上为止。

定理 4.15(完全性，$\Sigma\vDash\varphi\Rightarrow\Sigma\vdash\varphi$) 如果 φ 是模态逻辑语句集 Σ 的逻辑后承，那么 φ 有一个从 Σ 出发的模态表证明。

证明 设 φ 是 Σ 的逻辑后承。命题 4.10 的论证也证明了从 Σ 出发的 CSMT 是完成的。该定义还保证，对于任何 $\psi\in\Sigma$，$Tp\Vdash\psi$ 出现在任何含有 p 的不矛盾路径 P 上。定理4.11 的论证现在表明，如果根节点为 $Fp_1\Vdash\varphi$ 的从 Σ 出发的 CSMT 不是 φ 的从 Σ 出发的表证明，那么存在一个框架$\mathcal{C}$，使得每个 $\psi\in\Sigma$ 在其中被力迫，但 φ 在其中不被力迫。所以我们得到了所需要的矛盾。 □

值得一提的是，我们所定义的 CSMT 的特殊构造和应用 CSMT 对完全性所进行的论证说明，对于(模态逻辑中的)永真性而言，我们可以仅考虑每个可能世界只有有穷多个前驱的那些框架。其实，我们还可以要求可达关系的传递闭包具有这个有穷前驱性质。(见习题 5 ~ 7。)这些限制并不损害永真性概念，而对可达关系作其他的限制则有可能造成永真性概念的根本改变。这样的限制包括自反性和传递性，将在第五节中予以考虑。

习题

1. 针对情形(F$\forall$)和(F$\Box$)完成引理 4.5 的证明。
2. 对于联结词$\neg$，$\wedge$ 和$\rightarrow$，完成定理 4.11 证明的第(ii)部分。
3. 验证定理 4.11 的证明第(iii)部分中的存在量词情形。
4. 验证定理 4.11 的证明第(iv)部分中的$\Diamond$情形。
5. 证明：模态语句 φ 是永真的，当且仅当它在每个框架$\mathcal{C}=(W,\ S,\ \mathcal{C}(p))$中被力迫，其中 S 具有有穷前驱性质(finite predecessor property)，即对于任何 $p\in W$，集合 $\{q\in W\mid qSp\}$ 是有穷的。
6. 如果 S 是集合 W 上的二元关系，S 的传递一自反闭包(transitive-reflexive closure)TRC(S)，是 W 上所有包含 S 的自反且传递的二元关系 T 的交。证明：$\mathrm{TRC}(S)=\cup S_n$，其中 $S_0=S$，$S_1=S\cup\{(p,\ p)\mid\exists q(pSq\vee qSp)\}$，且 $S_{n+1}=S_n\cup\{(p,\ q)\mid\exists w(pS_nw\wedge wS_nq)\}$。
7. 证明：模态语句 φ 是永真的，当且仅当它在所有框架$\mathcal{C}=(W,\ S,\ \mathcal{C}(p))$中被力迫，其中对于每个 $p\in W$，集合 $\{q\in W\mid q\mathrm{TRC}(S)p\}$ 是有穷的。
8. **常论域**(constant domains)：我们可以限定所有可能世界具有相同论域，并以此改变我们对于模态逻辑的观念。在关于框架的定义 2.1 中，我们要求对于任何 p，$q\in W$ 有 $\mathcal{C}(p)=\mathcal{C}(q)$。力迫和永真性的定义

2.2 和定义 2.3 保持不变。对于表的定义 3.2，我们把(ii)中术语“任何适当的 c”的意思改变为任何在$\mathcal{L}$中或出现在 P 的某个表值中的 c。用这些定义证明常论域模态逻辑的可靠性与完全性定理。

第五节 模态公理和特殊的可达关系

对于模态逻辑许多的具体应用，采用不同特殊类型的可达关系似乎是适当的。例如，在分析计算机器行为时，你可能会认为可达关系应当反映机器所看到的时间特性。在这样的情形中要求可达关系具有自反性或传递性也许是适当的。如果你对不间断进程感兴趣，你可能会要求每个状态有一个严格的后继。从另一个观点来说，模态算子的一些特殊解释可能会提出一些你希望添加到模态逻辑中的公理。因此，例如，如果$\Box$的意思是“……必然为真”或“我知道……”，那么你也许想加入这样的公理模式$\Box\varphi\to\varphi$。另一方面，如果$\Box$的意思是“我相信……”，那么我们可能会拒绝把$\Box\varphi\to\varphi$ 作为公理：我可能有错误的信念。然而，在给信念建立模型的情形中，我们可能会采纳一个“自省”公理$\Box\varphi\to\Box\Box\varphi$(我所相信的，我相信我相信)或$\neg\Box\varphi\to\Box\neg\Box\varphi$(我不相信的，我相信我不相信)。但是，如果$\Box$的意思是“……必然为真”，那么后一个公理($\neg\Box\varphi\to\Box\neg\Box\varphi$)就不那么令人信服了。

已经证明，在给框架的可达关系所附加的一些自然限制与各种模态逻辑的一些共同的公理之间存在着紧密的联系。事实上，常常有可能两个方面所描述的恰好是同一个事物，具体来说就是，在具有某种特殊类型可达关系的所有框架中都被力迫的语句恰好就是某个公理系统的逻辑后承。本节给出了这一现象的几个例子。下一节，对于这些不同的模态逻辑，我们以第一章第七节和第二章第八节的样式描述传统希尔伯特式公理与规则系统。

在考虑具体例子之前，介绍几个一般的术语，以简化我们的讨论。

定义 5.1

*(i)令$\mathcal{F}$为一个框架类且 φ 为模态语言$\mathcal{L}$的语句。我们说 φ 是$\mathcal{F}$-**永真的**，记为$\vDash_{\mathcal{F}}\varphi$，如果 φ 在每个$\mathcal{C}\in\mathcal{F}$中被力迫。*

*(ii)令 F 为一个关于表展开的规则或规则族，即形如“如果 τ 是一个表，P 是 τ 上的路径且 τ' 由 τ 在 P 的末端添加某个(特殊类型的)表值 E 得到，那么 τ' 也是一个表”的规则或规则集。按照定义 3.2 的样子对 F-**表**进行归纳定义，其中定义 3.2 的(ii)要包括 F 中的构成规则。一个 F-表是某个语句 φ 的 F-表证明，如果它是有穷的，根节点形如 $Fp\Vdash\varphi$ 且每条路径都是矛盾的。我们说 φ 是 F-**可证的**(或 F-**表可证的**)，记为$\vdash_F\varphi$，如果它有一个 F-表证明。*

例 5.2(知识公理与自反性)　我们在例 3.6 中看到模式$\Box\varphi\to\varphi$ 不是永真的。这一模式习惯上称为*知识公理*并记为 T。如果你给知识建立模型，并且认为你不可能知道 φ，除非 φ 为真，那么这似乎就是一个合理的公理。你可能想要了解它的逻辑后承。有一个证明论的处理办法：我们可以只考虑由 T 的所有实例(的闭包)组成其前件集的表演绎。

语法上来说，这种办法很难操作。这很大程度地破坏了表方法相对于希尔伯特式系统而言的明显优势。主要问题是，如果我们要得到一个聪明的简短证明(而不是使用 CSMT)，我们没有好的办法可以知道该把公理的哪些实例放在任何特定的地方。然而，基本的表方法一般以相当直接且容易预测的方式给出证明。此外，公理化方法很少能让我们洞察到任何对应于 T 的语义。很难知道，要求 T 的每个实例在所考虑的每个框架中被力迫会对这个框架类给出什么样的刻画。

要解决这两个问题，可以通过(在例 3.6 中所做的)尝试证明 T 的某个典型实例：

$$
\begin{array}{lll}
1 & Fw \Vdash \Box\varphi \to \varphi & \\
 & | & \\
2 & Tw \Vdash \Box\varphi & \text{由}1 \\
 & | & \\
3 & Fw \Vdash \varphi & \text{由}1
\end{array}
$$

在这个地方，所尝试的证明陷入困境。然而显然，如果我们知道 wSw，那么我们将有个完整的证明。这建议我们把自反性加入到关于 T 的语义和表证明规则中。

定义 5.3

(i)$\mathcal{R}$是所有**自反框架**(reflexive frame)类，即其中所有框架的可达关系是自反的(对每个 $w \in W$ 有 wSw)。

(ii)R 是一个**自反表展开规则**(reflexive tableau development rule)指，给定表 τ，我们可以在 τ 中含有 w 的任何路径 P 的末端加上 wSw，从而形成新表 τ'。

(iii)T 是模式 T：$\Box\varphi\to\varphi$ 所有实例的全称闭包集。

就下列确切的意义而言，我们认为这些概念准确地抓住了 T 的要义：

定理 5.4 对于模态语言$\mathcal{L}$中任何语句 φ，下列条件是等价的：

(i)$T \vDash \varphi$，φ 是T的逻辑后承。

(ii)$T \vdash \varphi$，φ 是从T出发表可证的。

(iii)$\vDash_{\mathcal{R}} \varphi$，$\varphi$ 在每个自反的$\mathcal{L}$框架中被力迫。

(iv)$\vdash_{\mathcal{R}} \varphi$，$\varphi$ 是用自反表展开规则 R 可证的。

注意，这个定理不仅刻画了公理模式 T 的逻辑后承，它还包含了模态逻辑的可靠性(iv⇒iii)与完全性(iii⇒iv)定理，在该模态逻辑的语义中永真性意味着在所有自反框架中被力迫，而其证明系统由通常的构造规则加上 R 组成。(i)与(ii)等价就是从前件出发的演绎的可靠性与完全性定理(定理 4.13 和定理 4.15)。要完成这些等价关系的证明，我们只需证明关于 R 的可靠性与完全性结果(iii⇔iv)以及(i)与(iii)的等价性。

证明 可靠性，$\vdash_R \varphi \Rightarrow \vDash_{\mathcal{R}} \varphi$。设 τ 是一个根为 $Fp \Vdash \varphi$ 的 R－表且$\mathcal{C}=(W, S, \mathcal{C}(p))$是一个自反框架。如果存在 $p \in W$，使得 $p \nVdash \varphi$，那么定理 4.2 的证明表明$\mathcal{C}$与 τ 的某条路径一致。唯一的新情况是用规则 R 把某个表值 wSw 放在 τ 上时，根据 S 的自反性，它在$\mathcal{C}$中成立。因此，如果 τ 是 φ 的一个 R－证明($\vdash_R \varphi$)，那么就不能存在一个不力迫 φ 的自反框架，即$\vDash_{\mathcal{R}} \varphi$。

完全性，$\vDash_{\mathcal{R}}\varphi \Rightarrow \vdash_R \varphi$。我们从根为 $Fp_1 \Vdash \varphi$ 的完全系统自反模态表(complete systematic reflexive modal tableau，R-CSMT)开始。此概念的定义是显然的。它与定义 4.14 中从前件集 Σ 出发的 CSMT 的定义相同，除了在奇数阶段 $n=2k+1$，对于每个 $i<k$，把 p_iSp_i 接到每条含有 p_i 的不矛盾路径 P 上之外。如果由 $Fp_1 \Vdash \varphi$ 开始的 R-CSMT 不是 φ 的一个 R－证明，那么其中必有一条不矛盾的路径 P。命题 4.10 的论证再次表明任何 R-CSMT 是完成的。根据其定义易知 pSp 出现在每条含有 p 的不矛盾路径 P 上。因此，定理 4.11 的证明根据 P 定义了一个与 P 一致的自反框架$\mathcal{C}$。这就证明了 φ 不是$\mathcal{R}$-永真的。

$T \vDash \varphi \Rightarrow \vDash_{\mathcal{R}} \varphi$。由力迫$\Box\varphi$ 的定义 2.2 的(ix)立即可得 T 的每个实例在每个自反框架中都被力迫。因此，如果在每个使得 T 的所有实例被力迫的框架中 φ 都被力迫，则 φ 在每个自反框架中被力迫，即$T \vDash \varphi \Rightarrow \vDash_{\mathcal{R}} \varphi$。

另外还可以这样证明。记得在前面的例子中，T某个典型成员$\Box\varphi\to\varphi$ 的表证明尝试可

以很容易地变成一个 R-证明。（注意，$\mathcal{T}$的任一成员形如$\forall\vec{x}(\Box\varphi\to\varphi)$。在这种情况下，其表证明要首先用新常元替换全称量词所约束的变元。然后再像以前那样进行下去。）因此，对于每个$\theta\in\mathcal{T}$，有$\vdash_R\theta$。于是，由刚才证明的$\vdash_R$的可靠性定理知，对于每个$\theta\in\mathcal{T}$，有$\vDash_R\theta$。特别地，如果$\mathcal{T}\vDash\varphi$，则φ在每个自反框架中被力迫，即$\vDash_R\varphi$。（由于这个证明比较容易，下面例子中就采用这种证法。）

$\vDash_R\varphi\Rightarrow\mathcal{T}\vDash\varphi$。为了得出矛盾，设$\vDash_R\varphi$但$\mathcal{T}\nvDash\varphi$。令$\tau$是根为$Fp_1\Vdash\varphi$的从$\mathcal{T}$出发的 CSMT。根据假设与定理 4.13，$\tau$不是$\varphi$的一个证明，即在$\tau$中存在一条不矛盾路径$P$。令$\mathcal{C}=(W, S, \mathcal{C}(p))$是如同定理 4.11 中那样根据$P$而定义的框架。令$\mathcal{C}'=(W', S', \mathcal{C}'(p))$为$\mathcal{C}$的自反闭包，即$W'=W$，$\mathcal{C}'(p)=\mathcal{C}(p)$，而$S'=S\cup\{(w, w)\mid w\in W\}$是$S$的自反闭包。我们断言自反框架$\mathcal{C}'$与$P$一致。因此$p_1\nVdash_{\mathcal{C}'}\varphi$，从而得所需要的矛盾。 □

证明$\mathcal{C}'$与P一致与定理 4.11 中证明$\mathcal{C}$与P一致是相同的，除了在归纳步中还需要论证，由于$Tp\Vdash\Box\psi$或$Fp\Vdash\Diamond\psi$出现在P上，分别有$p\Vdash_{\mathcal{C}'}\psi$或$p\nVdash_{\mathcal{C}'}\psi$。显然这只需证明下面的引理（情形$p=q$是非平凡的，并且也是我们马上要用到的）：

引理 5.5

(i) 如果$Tp\Vdash\Box\psi$出现在P上且$pS'q$，那么$Tq\Vdash\psi$出现在P上。

(ii) 如果$Fp\Vdash\Diamond\psi$出现在P上且$pS'q$，那么$Fq\Vdash\psi$出现在P上。

证明 首先注意到，如果$p\neq q$，那么pSq出现在P上，根据τ是完成的事实可得出我们的结论。因此设$p=q$。

(i) 设$Tp\Vdash\Box\psi$出现在P上且ψ形如$\theta(c_1, \cdots, c_n)$，其中显示的c_i是ψ中所有不在初始语言中的常元。$\mathcal{T}$的元素$\forall x_1\cdots\forall x_n(\Box\theta(x_1, \cdots, x_n)\to\theta(x_1, \cdots, x_n))$出现在$P$上，因为$\tau$是一个来自$\mathcal{T}$的 CSMT。由于$\tau$是完成的且$Tp\Vdash\Box\theta(c_1, \cdots, c_n)$出现在$P$上，所以$Tp\Vdash\Box\theta(c_1, \cdots, c_n)\to\theta(c_1, \cdots, c_n)$也在$P$上出现。因此，$Tp\Vdash\Box\psi\to\psi$在$P$上出现。由于$\tau$是完成的，$Fp\Vdash\Box\psi$或$Tp\Vdash\psi$出现在$P$上。前者导致$P$矛盾。因此$Tp\Vdash\psi$出现在$P$上。

(ii) 如果$Fp\Vdash\Diamond\psi$出现在P上，那么（如同在(i)中那样）$Tp\Vdash\Box\neg\psi\to\neg\psi$出现在$P$上。因此，$Fp\Vdash\Box\neg\psi$或者$Tp\Vdash\neg\psi$出现在$P$上。在前一种情形中，通过把关于某个$w$的$pSw$与$Fw\Vdash\neg\psi$放在$P$上，我们最终可以在$P$上约化$Fp\Vdash\Box\neg\psi$。因为$\tau$是完成的，其中第一个$pSw$结合$Fp\Vdash\Diamond\psi$可以保证$Fw\Vdash\psi$出现在$P$上。类似地，第二个结合$Fw\Vdash\neg\psi$可以保证$Tw\Vdash\psi$出现在$P$上。因此在这种情形中，$P$是矛盾的，这与我们的假设矛盾。在后一种情形中，正如所要求的那样，$Fp\Vdash\psi$出现在P上。 □

至此完成定理 5.4 的证明。 □

例 5.6（自省和传递性） 模式 PI，$\Box\varphi\to\Box\Box\varphi$，过去习惯上称为"4"。现在经常称之为肯定性自省模式（scheme of positive introspection），因为它所表达的观点是：我所相信的，我相信我相信。我们再次看到，证明一个典型实例的尝试为我们找到适当的语义和证明规则提供了线索。（如图 51 所示。）

1	$Fw\Vdash\Box\varphi\to\Box\Box\varphi$	
2	$Tw\Vdash\Box\varphi$	由1
3	$Fw\Vdash\Box\Box\varphi$	由1
4	wSv 新的v	由3
5	$Fv\Vdash\Box\varphi$	由3
6	vSu 新的u	由5
7	$Fu\Vdash\varphi$	由5
8	$Tv\Vdash\varphi$	由2,4

图 51

该表证明没有矛盾。根据其中的原子语句，我们得到一个 3 世界框架$\mathcal{C}=(W, S, \mathcal{C}(p))$，其中$W=\{w, v, u\}$，$S=\{(v, u), (w, v)\}$，$\mathcal{C}(v)\vDash\varphi$且$\mathcal{C}(u)$，$\mathcal{C}(w)\nvDash\varphi$。这个框架$\mathcal{C}$是$\Box\varphi\to\Box\Box\varphi$永真性的反例，因为该语句在$\mathcal{C}$中的$w$上不被力迫。然而，我们给可达关系加上$wSu$之后可以产生一个矛盾。这里的关键是传递性。

定义 5.7

(i)$\mathcal{TR}$是所有**传递框架**(transitive frame)组成的类，即所有框架$\mathcal{C}=(W, S, \mathcal{C}(p))$，其中$S$是传递的：$wSv\wedge vSu\Rightarrow wSu$。

(ii)TR是**传递的表展开规则**(transitive tableau development rule)指，如果wSv和vSu出现在某表τ的路径P上，那么把wSu添加到P的末端可以得到另外一个表τ'。

(iii)$\mathcal{PI}$是模式 PI：$\Box\varphi\to\Box\Box\varphi$的实例的所有全称闭包集。

现在的情形很类似例 5.2。

定理 5.8 对于模态语言$\mathcal{L}$中任何语句φ，下列条件是等价的：

(i)$\mathcal{PI}\vDash\varphi$，$\varphi$是$\mathcal{PI}$的逻辑后承。

(ii)$\mathcal{PI}\vdash\varphi$，$\varphi$是从$\mathcal{PI}$出发表可证的。

(iii)$\vDash_{\mathcal{TR}}\varphi$，$\varphi$在每个传递$\mathcal{L}$框架中被力迫。

(iv)$\vdash_{TR}\varphi$，φ是可用传递的表展开规则证明的。

本定理的证明与关于自反框架的相应结果(定理 5.4)的证明是基本相同的。我们把自反、R、$\mathcal{R}$和$\mathcal{T}$分别替换为传递、TR、$\mathcal{TR}$和$\mathcal{PI}$，并在其他方面作相应的改动。例如，当存在某个q，使得p_iSq和qSp_j出现在P上时，在 TR-CSMT 的$n=2k+1$阶段把表值p_iSp_j接到每条不矛盾路径P的末端。与定理 5.4 的证明根本不同的唯一部分在于证明$\vDash_{\mathcal{TR}}\varphi\Rightarrow\mathcal{PI}\vDash\varphi$，特别地，证明类似于引理 5.5 的结果。

证明($\vDash_{\mathcal{TR}}\varphi\Rightarrow\mathcal{PI}\vDash\varphi$) 为了得到矛盾，设$\vDash_{\mathcal{TR}}\varphi$但$\mathcal{PI}\nvDash\varphi$。令$\tau$是根为$Fp_1\Vdash\varphi$的从$\mathcal{PI}$出发的 CSMT。根据假设和定理 4.13，$\tau$不是$\varphi$的证明。所以$\tau$中有一条不矛盾路径$P$。令$\mathcal{C}=(W, S, \mathcal{C}(p))$是如同定理 4.11 中那样根据$P$所定义的框架。令$\mathcal{C}'=(W', S', \mathcal{C}'(p))$为$\mathcal{C}$的传递闭包，即$W'=W$，$\mathcal{C}'(p)=\mathcal{C}(p)$，而$S'$是$S$的传递闭包，即$S'\equiv\cup S_n$，其中$S_0=S$且$S_{n+1}=S_n\cup\{(p, q)\mid\exists w(pS_nw\wedge wS_nq)\}$。(注意，根据其构造方法，$S'$是传递的。)我们断言传递框架$\mathcal{C}'$与$P$一致。因此$p_1\nVdash_{\mathcal{C}'}\varphi$，从而得到所需要的矛盾。 □

证明$\mathcal{C}'$与P一致和定理 4.11 中证明$\mathcal{C}$与P一致也是相同的，除了在归纳步中还需要论证，如果$Tp\Vdash\Box\psi$或$Fp\Vdash\Diamond\psi$出现在P上，则对于使得$pS'q$的每个q分别有$q\Vdash_{\mathcal{C}'}\psi$或$q\nVdash_{\mathcal{C}'}\psi$。显然这也只需证明下面的引理：

引理 5.9

(i)如果$Tp\Vdash\Box\psi$出现在P上且$pS'q$，那么$Tq\Vdash\psi$出现在P上。

(ii)如果$Fp\Vdash\Diamond\psi$出现在P上且$pS'q$，那么$Fq\Vdash\psi$出现在P上。

证明 对把(p, q)放入S'中的阶段n进行归纳证明。对于$n=0$，两个情形中的结论可由τ是完成的得到。设引理对$(u, v)\in S_n$和$(p, q)\in S_{n+1}$成立。令w使得(p, w)，$(w, q)\in S_n$。

(i)如果$Tp\Vdash\Box\psi$出现在P上，那么(如同引理 5.5 的论证)$Tp\Vdash\Box\psi\to\Box\Box\psi$也出现在$P$上。因为$\tau$是完成的，或者$Fp\Vdash\Box\psi$或者$Tp\Vdash\Box\Box\psi$出现在$P$上。因为$P$是不矛盾的，一定是$Tp\Vdash\Box\Box\psi$出现在$P$上。因为$(p, w)\in S_n$，根据归纳，$Tw\Vdash\Box\psi$出现在$P$上。又因为$(w, q)\in S_n$，所以$Tq\Vdash\psi$也出现在$P$上。 □

(ii)此种情形的证明是类似的，留作习题1。 □

例5.10(否定性自省和欧几里得框架) 我们下一个要考虑的模式是NI，否定性自省模式(negative introspection scheme)(我所不相信的，我相信我不相信)：$\neg\Box\varphi\rightarrow\Box\neg\Box\varphi$。这一模式习惯上记为“E”(来源于Euclidean)或“5”。图52给出证明NI一个实例的尝试。

完成这个证明的关键是当我们同时有wSv和wSu时可以添加uSv。然后就能由第9行推出跟第6行矛盾的$Tv\Vdash\varphi$。到目前为止我们对于这样的处理应当是十分熟悉的。

1	$Fw\Vdash\neg\Box\varphi\rightarrow\Box\neg\Box\varphi$		
2	$Tw\Vdash\neg\Box\varphi$		由1
3	$Fw\Vdash\Box\neg\Box\varphi$		由1
4	$Fw\Vdash\Box\varphi$		由2
5	wSv	新的v	由4
6	$Fv\Vdash\varphi$		由4
7	wSu	新的u	由3
8	$Fu\Vdash\neg\Box\varphi$		由3
9	$Tu\Vdash\Box\varphi$		由8

图 52

定义5.11

(i)$\mathcal{E}$是所有**欧几里得框架**(Euclidean frame)组成的类，即所有这样的框架$\mathcal{C}=(W, S, C(p))$，使得S是**欧几里得的**(Euclidean)：$wSv\wedge wSu\Rightarrow uSv$。

(ii)E是**欧几里得表展开规则**(Euclidean tableau development rule)指，如果wSv与wSu出现在表τ的路径P上，那么把uSv添加到P的末端可产生另外一个表τ'。

(iii)$\mathcal{NI}$是模式NI：$\neg\Box\varphi\rightarrow\Box\neg\Box\varphi$的实例的所有全称闭包集。

定理5.12 对于模态语言$\mathcal{L}$中任何语句φ，下列条件是等价的：

(i)$\mathcal{NI}\vDash\varphi$，$\varphi$是$\mathcal{NI}$的逻辑后承。

(ii)$\mathcal{NI}\vdash\varphi$，$\varphi$是从$\mathcal{NI}$出发表可证的。

(iii)$\vDash_{\mathcal{E}}\varphi$，$\varphi$在每个欧几里得$\mathcal{L}$框架中被力迫。

(iv)$\vdash_{E}\varphi$，φ是用欧几里得表展开规则可证的。

这个定理的证明与关于自反或传递框架的对应结果(定理5.4和引理5.9)的证明基本相同。证明$\vDash_{\mathcal{E}}\varphi\Rightarrow\mathcal{NI}\vDash\varphi$所需要的框架$\mathcal{C}'$是$\mathcal{C}$的欧几里得闭包(Euclidean closure)。其定义为在$\mathcal{C}$的基础上令$S'=\cup S_n$，其中$S_0=S$且$S_{n+1}=\{(p, q)\mid\exists w((w, p), (w, q)\in S_n)\}$。关键还是证明一个与引理5.5相对应的结果：

引理5.13

(i)如果$Tp\Vdash\Box\psi$出现在P上且$pS'q$，那么$Tq\Vdash\psi$出现在P上。

(ii)如果$Fp\Vdash\Diamond\psi$出现在P上且$pS'q$，那么$Fq\Vdash\psi$出现在P上。

证明 对把(p, q)放入S'中的阶段n进行归纳。对于$n=0$，两个情形中的结论可由τ是完成的得到。设引理对$(u, v)\in S_n$和$(p, q)\in S_{n+1}$成立。令w使得(w, p)，$(w, q)\in S_n$。

(i)如果$Tp\Vdash\Box\psi$出现在P上，那么(同以前一样)$Tw\Vdash\neg\Box\psi\rightarrow\Box\neg\Box\psi$也出现在$P$上。因为$\tau$是完成的，$Fw\Vdash\neg\Box\psi$或者$Tw\Vdash\Box\neg\Box\psi$出现在$P$上。在前一情形中，$Tw\Vdash\Box\psi$出现在$P$上，且根据归纳，正如所需要的那样，$Tq\Vdash\psi$也出现在$P$上。在后一情形中，$Tp\Vdash\neg\Box\psi$会出现在$P$上。因为这会导致$Fp\Vdash\Box\psi$出现在$P$上，从而使得$P$是矛盾的，这与我们的假设矛盾。因此，$Fw\Vdash\neg\Box\psi$，从而得$Tq\Vdash\psi$出现在$P$上。

(ii)此种情形的证明是类似的，留作习题2。 □

例5.14(序列公理和框架) 下一个例子是传统上称为D的模式：$\Box\varphi\rightarrow\neg\Box\neg\varphi$。它表

示如果我相信 φ，那么我不相信 $\neg\varphi$。现在常称之为序列模式(serial scheme)。我们还是从尝试证明 D 的某个实例开始：

1	$Fw \Vdash \Box\varphi \to \neg\Box\neg\varphi$	
2	$Tw \Vdash \Box\varphi$	由1
3	$Fw \Vdash \neg\Box\neg\varphi$	由1
4	$Tw \Vdash \Box\neg\varphi$	由3

图　53

空可达关系的框架提供了一个反例。假如存在一个 v，使得 wSv 的话，我们就能够推出所需要的矛盾。(我们将可以从($T\Box$)推出 $Tv \Vdash \varphi$，从($T\Box$)和($F\neg$)推出 $Fv \Vdash \varphi$。)

定义 5.15

(i)$\mathcal{SE}$上所有**序列框架**(serial frame)类，即所有这样的框架$\mathcal{C}=(W, S, \mathcal{C}(p))$，使得对于每个 $p \in W$，存在 q，使得 pSq。

(ii)SE 是**序列表展开规则**(serial tableau development rule)指，如果 p 出现在表 τ 的路径 P 上，那么对于某个新 q，把 pSq 添加到 P 的末端可产生另外一个表 τ'。

(iii)$\mathcal{D}$是模式 D：$\Box\varphi \to \neg\Box\neg\varphi$ 的实例的所有全称闭包集。

定理 5.16　对于模态语言$\mathcal{L}$中任何语句 φ，下列条件是等价的：

(i)$\mathcal{D} \vDash \varphi$，$\varphi$ 是$\mathcal{D}$的逻辑后承。

(ii)$\mathcal{D} \vdash \varphi$，$\varphi$ 是从$\mathcal{D}$出发的表可证的。

(iii)$\vDash_{\mathcal{SE}} \varphi$，$\varphi$ 在每个序列$\mathcal{L}$框架中被力迫。

(iv)$\vdash_{\mathrm{SE}} \varphi$，$\varphi$ 是用序列表展开规则可证的。

证明　证明同以前一样。这里的关键是，在验证$\vDash_{\mathcal{SE}} \varphi \Rightarrow \mathcal{D} \vDash \varphi$ 时，根据从$\mathcal{D}$出发的 CSMTτ 上的路径 P 所定义的框架$\mathcal{C}$必然是序列的。如果 p 出现在 P 上，那么，对于某个 ψ，$Tp \Vdash \Box\psi \to \neg\Box\neg\psi$ 也出现在 P 上。因为 τ 是完成的，$Fp \Vdash \Box\psi$ 或者 $Tp \Vdash \neg\Box\neg\psi$ 出现在 P 上。不管哪一种情形，应用 τ 是完成的事实，可知在 P 此后的展开中必定存在一个表值 pSq，其中 q 是新的，从而$\mathcal{C}$是序列的。 □

关于表规则及其对应框架类的其他例子可见习题 3 ~ 4 和第六节习题 1 ~ 4。

习题

1. 证明引理 5.9(ii)。
2. 证明引理 5.13(ii)。
3. **树形框架**：一个框架$\mathcal{C}$称为树形框架(tree frame)，如果按照规则“$p<q$ 当且仅当 pSq”其可达关系 S 在 W 上定义了一棵(第一章定义 1.1 意义下的)树。证明：在定理 5.8 中相互等价的各条件中可以加上这一条：
(v)φ 在每个树形框架中被力迫。(提示：考虑第四节习题 5 ~ 7。)
4. **线性框架**：一个框架$\mathcal{C}$称为线性框架(linear frame)，如果按照规则“$p<q$ 当且仅当 pSq”其可达关系 S 在 W 上定义了一个线性序。列表规则 L 指，如果可能世界 w 和 v 出现在 τ 的路径 P 上，那么把新表(如图 54 所示)接到 P 的末端可形成另外一个表 τ'。证明：模态语言$\mathcal{L}$的语句 φ 在所有线性框架中永真，当且仅当它是用列表规则可证的。

图　54

*第六节　公理化方法

在本章最后描述模态逻辑的一个基本的希尔伯特式系统，并给出一些标准模态系统及其传统术语的简要目录。在第二章第八节中，所使用的语言仅含有联结词 $\neg$ 和 $\to$ 以及量词 $\forall$。与此一致，在这里我们仅使用模态算子$\Box$，并把$\Diamond$视为一个被定义的符号，即用 $\neg\Box\neg\varphi$ 取

代$\Diamond\varphi$。

第二章第八节中给出的经典逻辑证明系统，可以通过添加一个新公理模式与一个新规则而扩展为一个称为 K 的模态逻辑证明系统：

公理模式 6.1 (vi)$\Box(\alpha\to\beta)\to(\Box\alpha\to\Box\beta)$。

规则 6.2 (iii)**必然性**(necessitation)：从 α 推出$\Box\alpha$。

当然现在这个系统包含了原来的公理模式(i)～(v)以及(vi)在模态语言$\mathcal{L}$中的所有实例。类似地，那些规则可以应用到$\mathcal{L}$的所有公式上。

这里“证明”和“从语句集 Σ 出发的证明”，其定义与经典逻辑的一样。易知 K 的公理的实例都是永真的。(例 3.5 给出了(vi)的一个典型实例的表证明。其他公理在$\mathcal{L}$中的实例的经典表证明对于 K 而言仍然是正确的。注意，含自由变元的公式的永真性是指其全称闭包是永真的。)因为 K 的规则保持永真性，所以此系统的所有定理都是永真的。(注意，一旦我们记得 α 的永真性意味着它的全称闭包在每个框架中被力迫，就立刻可得 $\alpha\vDash\Box\alpha$ 即使对于含自由变元的 α 也是成立的。或者如同第三节习题 11 那样，构造 $\forall\vec{x}\alpha(\vec{x})\Vdash\forall\vec{x}\Box\alpha(\vec{x})$的表证明。)K 的一个标准的完全性定理表明，这些定理与我们的表系统所定义的定理是一样的。

我们在第五节所分析的模式常被添加到 K 上，以得到用于特殊研究的公理系统。下面所列的是这些希尔伯特式系统的完全性定理，见于 Hughes 和 Cresswell[1984，4.4]的第九章。因此它们与第五节中对应的系统具有相同的定理。

系统 T 包含 K 和例 5.2 中的公理模式 T：$\Box\varphi\to\varphi$。它通常被视为知识逻辑。正如我们所看到的，它的定理恰好是在所有自反框架中被力迫的那些语句。

系统 S4 包含 T 和例 5.6 的模式 PI：$\Box\varphi\to\Box\Box\varphi$。结合定理 5.4 与定理 5.8 的结果，同样可知 S4 的定理恰好是在每个自反和传递的框架中被力迫的那些语句(习题 1)。如果省略 T，那么我们得到包含 K 和模式 PI 的系统 K4。

系统 S5 包含 S4 和例 5.10 的模式 E：$\neg\Box\varphi\to\Box\neg\Box\varphi$。其定理恰好是在每个自反的、传递的欧几里得框架中被力迫的那些语句。因为自反的、传递的欧几里得关系恰好是等价关系(习题2)，S5 的定理是在每个可达关系为等价关系的框架中被力迫的那些语句(习题 3)。省略 T 可得包含 K4 和模式 E 的系统 K5。

事实上，对于 S5 我们可以再作一些的说明。其定理是在所有完全框架(complete frame)中被力迫的语句。

定义 6.3

(i)$\mathcal{M}$是所有**完全框架**类，即所有这样的框架$\mathcal{C}=(W, S, \mathcal{C}(p))$，其中对于每个 p，$q\in W$，有 pSq。

(ii)M 是**完全表展开规则**(complete tableau development rule)指，如果 p 和 q 出现在表 τ 的路径 P 上，那么添加上 pSq 可得另外一个表 τ'。

(iii)$\mathcal{S}5$ 是模式 T，PI 和 NI 的所有实例的全称闭包集。

定理 6.4 对于模态语言$\mathcal{L}$中任何语句 φ，下列条件是等价的：

(i)$\mathcal{S}5\vDash\varphi$，$\varphi$ 是$\mathcal{S}5$ 的逻辑后承。

(ii)$\mathcal{S}5\vdash\varphi$，$\varphi$ 是从$\mathcal{S}5$ 出发表可证的。

(iii)$\vDash_{\mathcal{M}}\varphi$，$\varphi$ 在每个完全的$\mathcal{L}$框架中被力迫。

(iv) $\vdash_M \varphi$，φ 是用完全表展开规则可证的。

当然，这个定理可像以前那样证明(习题4)。它还可以用习题5所提供的约化方法，由(习题3)关于等价关系框架的结果推出(习题6)。

类似这样的系统已被用作自认知逻辑(autoepistemic logic)的基础(Moore[1984，5.5]和[1985，5.5])。若扩展到多模态算子$\Box_A$(每个处理器 A 有一个)，它们适合于分布式代理网络的推理(Fagin et al. [1995，5.6])。

习题

1. 证明：S4 的定理，即$\mathcal{T}$和$\mathcal{TR}$的并的(逻辑)后承，恰好是在使用自反与传递两个表规则的系统中可证明的那些定理。这些也是在每个自反和传递框架中被力迫的语句。
2. 二元关系 S 是等价关系(equivalence relation)，如果它是自反的(wSw)、对称的($wSv \Rightarrow vSw$)和传递的($uSv \wedge vSw \Rightarrow uSw$)。证明：一个自反的、传递的二元关系 S 是等价关系，当且仅当它是欧几里得的。
3. 证明：S5 的定理，即$\mathcal{T}$，$\mathcal{TR}$和$\mathcal{NI}$的并的(逻辑)后承，是在使用自反、传递与欧几里得等表规则的系统中可证明的那些定理。这些也是在可达关系为等价关系的每个框架中被力迫的语句。
4. 给出定理 6.4 的一个直接证明。
5. 令$\mathcal{C}=(W, S, \mathcal{C}(p))$为一个框架，其中 S 是等价关系。对于 $w \in W$，令$[w]=\{p \in W \mid pSw\}$为 w 的等价类。如果$\mathcal{C}_w$为$\mathcal{C}$在$[w]$上的限制，即框架($[w]$，$S \upharpoonright [w] \times [w]$，$\mathcal{C}(p)$)，那么对于每个语句 φ，有$w \Vdash_{\mathcal{C}} \varphi \Leftrightarrow w \Vdash_{\mathcal{C}_w} \varphi$。
6. 应用习题 3 和 5 给出定理 6.4 的另一个证明。

进一步阅读建议

Hughes 和 Cresswell[1984，4.4]和 Chellas[1980，4.4]是两本不错的模态逻辑入门教材。后者是计算机科学基础文献中常用的参考书。比较新的入门书是 Popkorn[1994，4.4]。更高级的教材是 van Benthem[1983，4.4]；van Benthem[1988，4.4]和 Goldblatt[1993，4.4]也很有用。Fitting[1983，4.4]对模态逻辑表模型作了百科全书式的处理。特别是，在这最后一本书的第八章第七节中包含了命题模态逻辑可判定性的一个证明。关于语法和语义的许多不同定义也可在那里找到。

Linsky[1971，4.4]是模态逻辑早期重要的论文集，其中包括克里普克的开创性工作和具有哲学观点的一些片断。在 Nerode[1991，4.4]中有关于模态逻辑应用的一些其他背景信息和更全面的参考书目。

Galton[1987，5.6]，Goldblatt[1982，5.6]和[1992，5.6]和 Turner[1984，5.6]都重视不同模态逻辑在计算机科学中的应用。Fagin et al. [1995，5.6]是一篇关于知识与信念逻辑的综述。Thayse[1989，5.6]详尽地论述了面向演绎数据库和人工智能的模态逻辑。Thayse[1991，5.6]提供了这些观点在人工智能许多领域中的广泛应用。

第五章　直觉主义逻辑

第一节　直觉主义与构造主义

在过去的一个世纪中，数学哲学中的一个主要争论是如何看待数学中非能行或非构造的证明。宣称已经证明存在具有某性质的数，但是即使在原则上也不能实际地产生这样的数，这是否合理呢？宣称已证明存在某个函数而没有提供计算它的任何方法，这又是否合理呢？布劳威尔(L. E. J. Brouwer)也许是极端构造主义观点的最著名的早期倡导者。他拒绝20世纪早期数学的很多结果，就是因为这些结果没有提供可接受的存在性证明。他主张 $p \vee q$ 的一个证明必须要么是 p 的证明要么是 q 的一个证明，而 $\exists x P(x)$ 的证明必须包含证据 c 的构造和 $P(c)$ 为真的证明。大多数非构造性证明的核心是使用排中律：对于每个语句 A，$A \vee \neg A$ 为真。基于经典逻辑的这一定律，$\exists x P(x)$ 的证明可通过证明其否定命题导致矛盾来实现，而不用提供任何找到满足 P 的 x 的线索。类似地，可以通过证明 $\neg(\neg p \wedge \neg q)$ 来证明 $p \vee q$，而不用知道 p 和 q 哪一个为真。

例 1.1　我们希望证明存在两个无理数 a 和 b，使得 a^b 为有理数。令 $c = \sqrt{2}^{\sqrt{2}}$。如果 c 是有理数，那么取 $a = \sqrt{2} = b$。另一方面，如果 c 是无理数，那么 $c^{\sqrt{2}} = 2$ 是有理数，从而可取 $a = c$ 而 $b = \sqrt{2}$。因此，不管哪一种情况，我们都有两个无理数 a 和 b，使得 a^b 为有理数。这个证明依赖于排中律，因为我们设 c 要么是有理数要么不是。它没给出任何线索来确定这两对数中哪一对是所要求的。

例 1.2　考虑库尼西引理(第一章定理 1.4)的证明。我们曾用归纳法定义无穷路径。根据归纳，我们知道：在每一步的有限多的直接后继之中，必有一个后继的下面有无穷多个节点。然后“取出”这样的后继作为路径的下一个节点。我们已通过归纳法证明了一个析取项为真，因而接着采用了“分情形”的论证。由于我们没有用任何方式确定哪一个后继下面有无穷多个节点，故没有实际构造(没有用算法去定义)那条我们证明其存在的无穷路径。类似的考虑适用于我们关于完全性、紧致性和其他定理的证明。

一个试图把握布劳威尔的哲学立场的形式逻辑是由他的学生海丁(Heyting)所发展的。这个逻辑称为直觉主义逻辑(intuitionistic logic)。它是把握构造性推理的一个重要尝试。特别地，在直觉主义逻辑中排中律不再永真。

人们提出了一些范例用以解释布劳威尔的观点。这些范例中的每一个都可以为直觉主义逻辑提供模型或语义。有一个范例把数学命题视为关于我们(或某人)的知识或者所拥有的证明的断言。一个语句为真仅当我们知道它为真或者在我们证明它之后。任何时候我们都不知道以后会有什么样的事实将被发现或证明。这个解释很适合于计算机科学中涉及数据库和程序验证的许多情形。就数据库而言，有一点需要指出，我们认为我们的知识一直在增长，所以新事实会添加进来，但旧事实不会被删除，也不会与新事实产生矛盾。这一观点看起来对于数学知识的进步而言是合理的，但在其他许多情形中它是不准确的。只要在事实上附加时间的印记，这个模型在大多数时候就仍然可以使用。因此数据库记录的是我们知道什么与

我们知道的时间。直觉主义模型适合于这种数据库的演绎。

就程序验证来说，直觉主义逻辑为构造性证明检验器和推理系统的发展提供了基础。其关键的思想是，根据布劳威尔的想法，存在性命题的证明需要构造其证据。类似地，证明对于每个 x，存在 y，使得 $P(x, y)$ 成立，需要构造一个从 x 值计算出 y 值的算法。这个逻辑系统的魅力是显然的。在实践层面上，现在有一些大规模系统可以为这样的断言(互动地)提供直觉主义的证明。这些系统能够得出函数的实际计算方法。这样我们就有了一个经过验证的算法，因为其存在性的证明其实就是该算法可以实际正确运行的证明。NUPRL 就是这样的一个系统，它是由康奈尔大学的 R. Constable [1986, 5.6]和其他人研制开发的。

本章阐述直觉主义逻辑的基本内容，包括由克里普克所发展的一种语义，它反映了海丁形式主义的“知识状态”解释。选择这种语义足以反映直觉主义推理，除了出于直观的考虑之外，还有这样的事实为依据，即下面的析取性(disjunction property)和存在性(existence property)成立：

如果$(\varphi \vee \psi)$是直觉主义永真的，那么 φ 或者 ψ 是直觉主义永真的。

如果$\exists x\varphi(x)$是直觉主义永真的，那么对于某个常元 c 来说 $\varphi(c)$也是直觉主义永真的。

然后，和经典逻辑一样，发展一个以表方法为基础的直觉主义证明论，并证明相应的可靠性和完全性定理。当然，完全性定理把上面的定理转变为关于可证明性的定理。我们能(直觉主义地)证明 $\varphi \vee \psi$ 仅当我们能证明其中之一。我们能证明 $\exists x\varphi(x)$仅当我们能对某个明确的常元 c 证明 $\varphi(c)$。

本章内容独立于第四章，因此二者有一些内容上的重复。我们在第六节中为已经看过第四章的读者提供了一个向导，用以比较经典的、模态的和直觉主义的逻辑。

第二节 框架和力迫

除了一处修改和两个限制之外，我们所用的语言与第二章中的经典谓词逻辑语言是一样的。这些修改和限制可以简化展开力迫时所需要的技术细节。其中修改是指，在形式上从我们的语言中省略了逻辑联结词$\leftrightarrow$，而把 $\varphi\leftrightarrow\psi$ 视为$(\varphi\rightarrow\psi)\wedge(\psi\rightarrow\varphi)$的缩写。其中的限制在于我们语言的非逻辑的组成部分。我们假设在整个这一章中每个语言$\mathcal{L}$至少有一个常元符号，但除了常元之外没有函数符号。

我们现在给出直觉主义逻辑的一个语义，来形式化“知识状态”解释。

定义 2.1 *令$\mathcal{C}=(R, \leqslant, \{\mathcal{C}(p)\}_{p\in R})$包含一个偏序集$(R, \leqslant)$和一个指派，将 R 中每个 p 指派一个$\mathcal{L}$的(在第二章定义 4.1 意义下的)结构$\mathcal{C}(p)$。为简化符号，我们把形式上更精确的$\mathcal{C}=(R, \leqslant, \{\mathcal{C}(p)\}_{p\in R})$记为$\mathcal{C}=(R, \leqslant, \mathcal{C}(p))$。与通常一样，用 $C(p)$表示结构$\mathcal{C}(p)$的论域。还用$\mathcal{L}(p)$表示$\mathcal{L}$的如下扩张：以第二章第四节中定义真值的方式对于 $C(p)$的每个元素 a 添加一个名字 c_a。$A(p)$表示在$\mathcal{C}(p)$中为真的$\mathcal{L}(p)$的原子公式集。我们称$\mathcal{C}$是***语言$\mathcal{L}$的一个框架***，或简称为$\mathcal{L}$**框架**，如果对 R 中每个 p 和 q，$p\leqslant q$ 蕴涵 $C(p)\subseteq C(q)$，$\mathcal{L}(p)\subseteq\mathcal{L}(q)$中常元的解释与其在$\mathcal{C}(p)$以及$\mathcal{C}(q)$中的解释是一样的，并且 $A(p)\subseteq A(q)$。*

*$p\leqslant q$ 常读作“q **扩展了**(extend)p”或者“q 是 p 的一个**未来**(future)”。R 的元素称为**力迫条件**(forcing condition)，**可能世界**(possible world)或**知识状态**(state of knowledge)。*

我们现在定义框架上的力迫关系。

定义 2.2(框架的力迫) 令$\mathcal{C}=(R, \leqslant, \mathcal{C}(p))$为语言$\mathcal{L}$的一个框架，$p$在$R$中且$\varphi$是语言$\mathcal{L}(p)$中的语句。对语句$\varphi$作归纳，我们定义$p$ **力迫** φ，记为$p \Vdash \varphi$。

(i) 对于原子语句φ，$p \Vdash \varphi \Leftrightarrow \varphi$在$A(p)$中。

(ii) $p \Vdash (\varphi \to \psi) \Leftrightarrow$对于所有$q \geqslant p$，$q \Vdash \varphi$蕴涵$q \Vdash \psi$。

(iii) $p \Vdash \neg\varphi \Leftrightarrow$对于所有$q \geqslant p$，$q$不力迫$\varphi$。

(iv) $p \Vdash (\forall x)\varphi(x) \Leftrightarrow$对于每个$q \geqslant p$和$\mathcal{L}(q)$中的每个常元$c$，$q \Vdash \varphi(c)$。

(v) $p \Vdash (\exists x)\varphi(x) \Leftrightarrow \mathcal{L}(p)$中存在一个常元$c$，使得$p \Vdash \varphi(c)$。

(vi) $p \Vdash (\varphi \wedge \psi) \Leftrightarrow p \Vdash \varphi$且$p \Vdash \psi$。

(vii) $p \Vdash (\varphi \vee \psi) \Leftrightarrow p \Vdash \varphi$或$p \Vdash \psi$。

如果要指明所用的框架，我们可以说在$\mathcal{C}$中p力迫φ并记为$p \Vdash_{\mathcal{C}} \varphi$。

定义 2.3 令φ为语言$\mathcal{L}$中的语句。我们说φ **在$\mathcal{L}$框架$\mathcal{C}$中被力迫**，记为$\Vdash_{\mathcal{C}}\varphi$，如果$R$中每个$p$力迫$\varphi$。我们说$\varphi$是**直觉主义永真的**(intuitionistically valid)，如果φ在每个$\mathcal{L}$框架中被力迫。

分别定义$p \Vdash \varphi \to \psi$，$p \Vdash \neg\varphi$和$p \Vdash (\forall x)\varphi(x)$的(ii)，(iii)和(iv)，每个都有一个在偏序“$\leqslant$”的元素上取值的量词，即“对于所有q，如果$q \geqslant p$，那么……”。(ii)说，p力迫一个蕴涵式$\varphi \to \psi$，仅当任何更大的知识状态q，如果q力迫前件φ，那么q也力迫后承ψ。这表示在面对更多知识时蕴涵式的持续性。(iii)说，当没有比p更大的知识状态力迫φ时，p就力迫φ的否定。这说明如果φ不能被比p所提供的知识更多的知识所力迫，那么$\neg\varphi$就被力迫。(iv)说，p力迫一个全称语句，仅当该语句的所有实例在所有更大的知识状态中被力迫。这表示在p所提供之外的任何新知识面前，力迫全称语句的持久性。下面的引理表明力迫持久性的另外一个方面，即力迫不把过去计算在内，只计算将来。(注意，本章所使用的元语言(metalanguage)的逻辑仍然是经典的。例如，在(ii)中“蕴涵”的含义与第一章相同。)

引理 2.4(限制引理) 令$\mathcal{C}=(R, \leqslant, \{\mathcal{C}(p)\}_{p \in R})$为一个框架，令$q$在$R$中且$R_q=\{r \in R \mid r \geqslant q\}$。那么

$$\mathcal{C}_q=(R_q, \leqslant, \mathcal{C}(p))$$

是一个框架，其中“$\leqslant$”与函数$\mathcal{C}(p)$被限制在R_q上。此外，对于R_q中的r，在$\mathcal{C}$中r力迫φ，当且仅当在$\mathcal{C}_q$中r力迫φ。

证明 对公式长度应用归纳法进行证明，留作习题7。 □

考虑某个$\mathcal{L}$框架$\mathcal{C}$中的经典结构$\mathcal{C}(p)$。当我们从p走向一个大于p的q(即$q>p$)时，我们从p的经典结构$\mathcal{C}(p)$走向q的(可能)更大的经典结构$\mathcal{C}(q)$，其中有更多的原子语句经典地为真，从而更少的原子语句经典地为假。在(i)、(v)、(vi)和(vii)中，对于原子语句而言，“与”、“或”和“存在”的含义分别与第二章定义4.3中$\mathcal{C}(p)$上真值定义中的含义相同。对于其他的条款而言，有一个新的特点，事实上φ在$\mathcal{C}(p)$中的经典真值与p力迫φ并非总是一致的。然而，它们在一种特殊情形下确实是一致的。

引理 2.5(退化引理) 令$\mathcal{C}$为某语言$\mathcal{L}$的框架且φ是$\mathcal{L}$的语句。如果p是$\mathcal{C}$上偏序集R的极大元，那么φ在$\mathcal{C}(p)$中经典地为真，即$\mathcal{C}(p) \vDash \varphi$，当且仅当$p \Vdash \varphi$。特别地，如果$R$中仅有一个知识状态$p$，那么$\mathcal{C}(p) \vDash \varphi$当且仅当$p \Vdash \varphi$。

证明 对公式进行归纳证明。对于R的极大元p，$p \Vdash \varphi$定义中的各条款恰好与第二章

定义 4.3 中定义$\mathcal{C}(p)\vDash\varphi$的各条款吻合。在(ii)、(iii)和(iv)中对于未来知识状态的依赖性可以简单地归约为p上的经典情形。例如，考虑(ii)：$p\Vdash(\varphi\to\psi)\Leftrightarrow(\forall q\geqslant p)(q\Vdash\varphi$ 蕴涵 $q\Vdash\psi)$。因为p在R中是极大的，所以$q\geqslant p$与$q=p$等同。因此(ii)归约为($p\Vdash\varphi\to\psi$，当且仅当$p\Vdash\varphi$ 蕴涵 $p\Vdash\psi$)，这与对应的关于经典蕴涵的条款(第二章定义 4.3(v))相吻合。我们把验证所有其他条款的等价性留作习题 8。 □

定理 2.6 任何直觉主义永真的语句都是经典永真的。

证明 根据退化引理(引理 2.5)，每个经典模型是含单元素偏序集的框架模型，其中力迫与经典真值是等价的。由于一个语句如果在所有经典模型中为真它就是经典永真的，所以当它在每个框架中都被力迫时它是经典永真的。 □

接下来，再看一看经典永真的语句中哪些是直觉主义永真的和哪些不是。我们通过构造框架反例来说明如何验证一些经典永真语句不是直觉主义永真的。在给出例子之前，我们建立一些用于表示框架的符号约定。以下所有例子中的序都是完全二叉树的子序。这样我们可把相应的框架视为标签二叉树，其中节点p的标签为结构$\mathcal{C}(p)$，或者等价地是由$C(p)$和$A(p)$所构成的二元组。我们因此以通常的方式把框架画成标签二叉树且各标签的表示形如$\langle C(p), A(p)\rangle$。表方法的理论发展及其完全性的证明需要更一般的树，我们把它留到下一节再说。

在以下关于非直觉主义永真语句的例子中(例 2.7 ~ 例 2.11)，φ和ψ表示$\mathcal{L}$中的原子公式，它们不含有自由变元或者仅仅显示出来的x是自由变元。在这些例子中，偏序的底部节点$\varnothing$所对应的结构$\mathcal{C}(\varnothing)$，是$\mathcal{L}$中所有解释为c的常元所构成的集合C。我们从典型的经典永真但不是直觉主义永真的语句开始。

例 2.7 正如所期望的那样，语句$\varphi\vee\neg\varphi$(排中律的一个实例)不是直觉主义永真的。令框架$\mathcal{C}$为

$$\begin{array}{c}\langle C,\{\varphi\}\rangle\\ |\\ \langle C,\varnothing\rangle\end{array}$$

(因此我们把C同时当作框架的两个节点$\varnothing$和 0 的论域。)在底部节点上，没有原子事实为真，即$A(\varnothing)$为空。在顶端节点 0 上，令$A(0)=\{\varphi\}$，从而有单个原子事实φ为真。

现在考虑是否有$\varnothing\Vdash\varphi\vee\neg\varphi$。$\varnothing$肯定不力迫$\varphi$，因为$\varphi$是原子的且在$\mathcal{C}(\varnothing)$中不真，即不在$A(\varnothing)$中。另一方面，$0\Vdash\varphi$，因为$\varphi\in A(0)$。因此$\varnothing$不力迫$\neg\varphi$，因为它有一个扩展 0 力迫$\varphi$。所以根据定义，$\varnothing$不力迫$\varphi\vee\neg\varphi$且该语句不是直觉主义永真的。

例 2.8 语句$(\neg\varphi\to\neg\psi)\to(\psi\to\varphi)$不是直觉主义永真的。令框架$\mathcal{C}$为

$$\begin{array}{c}\langle C,\{\varphi,\psi\}\rangle\\ |\\ \langle C,\{\psi\}\rangle\end{array}$$

为了得到矛盾，设$\varnothing\Vdash(\neg\varphi\to\neg\psi)\to(\psi\to\varphi)$。那么根据力迫定义(定义 2.2)的(ii)，$\varnothing\Vdash(\neg\varphi\to\neg\psi)$蕴涵$\varnothing\Vdash(\psi\to\varphi)$。现在根据该定义的(iii)，$\varnothing$和 0 都不力迫$\neg\varphi$，因为$\varphi$在$A(0)$中，从而在 0 上被力迫。因此我们看到，再次应用(ii)以及只有$\varnothing$与 0 是$\geqslant\varnothing$的事实，$\varnothing$确实力迫$(\neg\varphi\to\neg\psi)$。另一方面，$\varnothing$不力迫$(\psi\to\varphi)$，因为$\varnothing$力迫ψ但不力迫φ，从而得到所需要的矛盾。

例 2.9 语句$(\varphi\to\psi)\vee(\psi\to\varphi)$不是直觉主义永真的。令框架$\mathcal{C}$为

$$\begin{tikzcd}[column sep=small]
\langle C,\{\varphi\}\rangle \arrow[dr, dash] & & \langle C,\{\psi\}\rangle \arrow[dl, dash] \\
& \langle C,\varnothing\rangle &
\end{tikzcd}$$

在此框架中，$\varnothing$既不力迫φ又不力迫ψ，0 力迫φ但不力迫ψ，且 1 力迫ψ但不力迫φ。由于在$\varnothing$之上有一个节点，即 0，它力迫φ但不力迫ψ，所以$\varnothing$不力迫$\varphi\to\psi$。类似地，$\varnothing$不力迫$\psi\to\varphi$。所以$\varnothing$不力迫$(\varphi\to\psi)\vee(\psi\to\varphi)$。

例 2.10 语句$\neg(\forall x)\varphi(x)\to(\exists x)\neg\varphi(x)$不是直觉主义永真的。令$b$为不同于$C$中唯一元素$c$的任何事物。令框架$\mathcal{C}$为

$$\begin{tikzcd}
\langle\{b,c\},\{\varphi(c)\}\rangle \arrow[d, dash] \\
\langle C,\varnothing\rangle
\end{tikzcd}$$

现在根据定义 2.2 的(iv)，$\varnothing$和 0 都不力迫$(\forall x)\varphi(x)$，因为$b\in C(0)$但 0 不力迫$\varphi(b)$。因此$\varnothing\Vdash\neg(\forall x)\varphi(x)$。如果$\varnothing\Vdash\neg(\forall x)\varphi(x)\to(\exists x)\neg\varphi(x)$（假如所给的语句永真的话该结论成立），那么$\varnothing$也会力迫$(\exists x)\neg\varphi(x)$。根据该定义的(v)这意味着存在一个$c\in C$，使得$\varnothing\Vdash\neg\varphi(c)$。因为$c$是$C$的唯一元素且$0\Vdash\varphi(c)$，所以$\varnothing$不力迫$(\exists x)\neg\varphi(x)$。

例 2.11 语句$(\forall x)(\varphi\vee\psi(x))\to\varphi\vee(\forall x)\psi(x)$不是直觉主义永真的。所需要的框架为

$$\begin{tikzcd}
\langle\{b,c\},\{\psi(c),\varphi\}\rangle \arrow[d, dash] \\
\langle C,\{\psi(c)\}\rangle
\end{tikzcd}$$

我们首先断言$\varnothing\Vdash(\forall x)(\varphi\vee\psi(x))$。因为$\varnothing\Vdash\psi(c)$和$0\Vdash\varphi$，结合关于析取式的(vii)与关于全称量词的(iv)，我们看到正如所断言的那样，$\varnothing\Vdash(\forall x)(\varphi\vee\psi(x))$。为了得出矛盾，现在设$\varnothing\Vdash(\forall x)(\varphi\vee\psi(x))\to\varphi\vee(\forall x)\psi(x)$。然后有$\varnothing\Vdash\varphi\vee(\forall x)\psi(x)$。然而$\varnothing$不力迫$\varphi$，且因为 0 不力迫$\psi(b)$，$\varnothing$也不力迫$(\forall x)\psi(x)$。因此$\varnothing$不力迫析取式$\varphi\vee(\forall x)\psi(x)$，所以得所需要的矛盾。

我们现在给出一些直觉主义永真的语句，其永真性可以直接用力迫定义验证。在给出例子之前，先证明一些关于力迫关系的基本事实，它们对于这些验证还有将来的论证都有用处。其中第一个也许是关于力迫的最有用的事实。它表达了在我们沿着偏序向上移动时力迫的稳定性。

引理 2.12（单调性引理） *对于$\mathcal{L}$的每个语句φ和每个$p,q\in R$，如果$p\Vdash\varphi$且$q\geqslant p$，那么$q\Vdash\varphi$。*

证明 我们根据φ的逻辑复杂性用归纳法证明引理。对于(i)，(ii)，(iii)和(iv)，在验证结论$q\Vdash\varphi$时不需要使用归纳假设。在第一种情形中，根据框架定义和定义原子语句力迫的(i)自身，立刻可得$q\Vdash\varphi$成立。其他的条款为了使得本引理成立分别精确地定义了（直觉主义的）蕴涵、否定和全称量词的含义。(v)、(vi)和(vii)分别定义存在量词、合取式和析取式的力迫，在这些条款的验证中我们使用归纳假设。

(i)如果φ是原子的且$p\Vdash\varphi$，那么φ在$A(p)$中。然而，根据框架的定义，$A(p)\subseteq A(q)$，从而φ在$A(q)$中。因此，根据定义，$q\Vdash\varphi$。

(ii)设$p\Vdash\varphi\to\psi$且$q\geqslant p$。要证明$q\Vdash\varphi\to\psi$，我们只需证明，如果$r\geqslant q$且$r\Vdash\varphi$，那么

$r \Vdash \psi$。现在根据传递性 $r \geqslant p$，由我们的假设 $p \Vdash \varphi \to \psi$ 和 $r \Vdash \varphi$，可得 $r \Vdash \psi$。

(iii) 设 $p \Vdash \neg\varphi$ 且 $q \geqslant p$。要证明 $q \Vdash \neg\varphi$，我们只需证明，如果 $r \geqslant q$，那么 r 不力迫 φ。根据传递性，$r \geqslant p$。再根据 $p \Vdash \neg\varphi$ 的定义，得 r 不力迫 φ。

(iv) 设 $p \Vdash (\forall x)\varphi(x)$ 且 $q \geqslant p$。要证明 $q \Vdash (\forall x)\varphi(x)$，我们只需证明，对任何 $r \geqslant q$ 与任何 $c \in C(r)$，有 $r \Vdash \varphi(c)$。根据传递性，$r \geqslant p$。然后根据 $p \Vdash (\forall x)\varphi(x)$ 的定义得，对于任何 $c \in C(r)$，$r \Vdash \varphi(c)$。

(v) 设 $p \Vdash (\exists x)\varphi(x)$ 且 $q \geqslant p$。那么根据力迫的定义，在 $C(p)$ 中存在 c，使得 $p \Vdash \varphi(c)$。根据归纳假设，$q \geqslant p$ 和 $p \Vdash \varphi(c)$ 蕴涵 $q \Vdash \varphi(c)$。因此 $q \Vdash (\exists x)\varphi(x)$。

(vi) 设 $p \Vdash (\varphi \wedge \psi)$ 且 $q \geqslant p$。那么根据力迫的定义，$p \Vdash \varphi$ 且 $p \Vdash \psi$。根据归纳假设，$q \Vdash \varphi$ 且 $q \Vdash \psi$。因此 $q \Vdash (\varphi \wedge \psi)$。

(vii) 设 $p \Vdash (\varphi \vee \psi)$ 且 $q \geqslant p$。那么根据力迫的定义，$p \Vdash \varphi$ 或 $p \Vdash \psi$。根据归纳假设，$q \Vdash \varphi$ 或 $q \Vdash \psi$。根据析取式力迫的定义，这就是说 $q \Vdash (\varphi \vee \psi)$。 □

单调性说明后来的知识状态 q 所增加的新原子语句不会改变前面知识状态的力迫。这个单调性特征，正如第三章第七节中所讨论的那样，把直觉主义框架中的“真理”与“非单调逻辑”中的“真理”区别开来。在非单调逻辑中，在知识状态 p 被力迫的语句不需要在知识状态 $q > p$ 被力迫。在框架中，随着时间的流逝，我们学到了新“事实”却从未发现虚假的旧“事实”。

引理 2.13(双重否定引理) *$p \Vdash \neg\neg\varphi$，当且仅当对于任何 $q \geqslant p$，存在 $r \geqslant q$，使得 $r \Vdash \varphi$。*

证明 $p \Vdash \neg\neg\varphi$，当且仅当任何 $q \geqslant p$ 不力迫 $\neg\varphi$，或等价地，当且仅当任何 $q \geqslant p$ 有 $r \geqslant q$，使得 $r \Vdash \varphi$。 □

引理 2.14(弱量词引理)

(i) $p \Vdash \neg(\exists x)\neg\varphi(x)$，当且仅当对于任何 $q \geqslant p$ 和任何 $c \in C(q)$，存在 $r \geqslant q$，使得 $r \Vdash \varphi(c)$。

(ii) $p \Vdash \neg(\forall x)\neg\varphi(x)$，当且仅当对于任何 $q \geqslant p$，存在 $s \geqslant q$ 和 $c \in C(s)$，使得 $s \Vdash \varphi(c)$。

证明 (i) 该结论根据定义立即可得。

(ii) $q \Vdash (\forall x)\neg\varphi(x)$，当且仅当对于任何 $r \geqslant q$ 和 $c \in C(r)$，不存在 $s \geqslant r$，使得 $s \Vdash \varphi(c)$。因此，q 不力迫 $(\forall x)\neg\varphi(x)$，当且仅当对存在 $r \geqslant q$ 和 $c \in C(r)$，使得对于某个 $s \geqslant r$，$s \Vdash \varphi(c)$。所以 $p \Vdash \neg(\forall x)\neg\varphi(x)$，当且仅当对于所有的 $q \geqslant p$，存在一个 $r \geqslant q$ 和一个 $c \in C(r)$，使得对于某个 $s \geqslant r$ 有 $s \Vdash \varphi(c)$。根据传递性，$s \geqslant q$ 且 c 在 $C(s)$ 中，结论得证。 □

我们现在给出所承诺的直觉主义永真的例子。在下列例子(例 2.15 ~ 例 2.19)中 φ 和 ψ 是任意语句。

例 2.15 $\varphi \to \neg\neg\varphi$ 是直觉主义永真的。为了证明任何 p 力迫 $\varphi \to \neg\neg\varphi$，我们设 $q \geqslant p$ 且 $q \Vdash \varphi$。我们必须证明 $q \Vdash \neg\neg\varphi$。根据双重否定引理，只要证明对于每个 $r \geqslant q$，存在 $s \geqslant r$，使得 $s \Vdash \varphi$。根据单调性引理，$r \Vdash \varphi$，从而 r 就是所需要的 s。

例 2.16 $\neg(\varphi \wedge \neg\varphi)$ 是直觉主义永真的。要证明任何 p 力迫 $\neg(\varphi \wedge \neg\varphi)$，我们需要证明没有 $q \geqslant p$ 力迫 $\varphi \wedge \neg\varphi$，或等价地没有 $q \geqslant p$ 既力迫 φ 又力迫 $\neg\varphi$。设 q 既力迫 φ 又力迫 $\neg\varphi$。现在 $q \Vdash \neg\varphi$ 意味着没有 $r \geqslant q$ 力迫 φ。由于 $q \geqslant q$，我们同时有 q 力迫 φ 和 q 不力迫 φ，矛盾。

例 2.17 $(\exists x)\neg\varphi(x)\to\neg(\forall x)\varphi(x)$ 是直觉主义永真的。要证明任何 p 力迫 $(\exists x)\neg\varphi(x)\to\neg(\forall x)\varphi(x)$，我们需要证明，如果 $q\geqslant p$ 且 $q\Vdash(\exists x)\neg\varphi(x)$，那么 $q\Vdash\neg(\forall x)\varphi(x)$。现在 $q\Vdash(\exists x)\neg\varphi(x)$ 说明在 $C(q)$ 中存在 c，使得 $q\Vdash\neg\varphi(c)$。根据单调性，任何 $r\geqslant q$ 也力迫 $\neg\varphi(c)$，所以没有这样的 r 力迫 $(\forall x)\varphi(x)$，因此 $q\Vdash\neg(\forall x)\varphi(x)$。这个例子应当与其逆命题(例 2.10)相比较，其逆命题是经典地永真但不是直觉主义地永真。

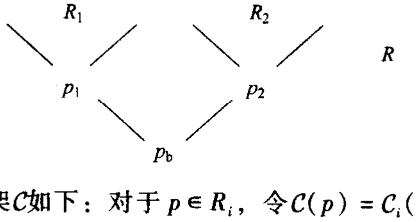
例 2.18 $\neg(\exists x)\varphi(x)\to(\forall x)\neg\varphi(x)$ 是直觉主义永真的。要证明任何 p 力迫 $\neg(\exists x)\varphi(x)\to(\forall x)\neg\varphi(x)$，我们必须证明，对于任何 $q\geqslant p$，如果 $q\Vdash\neg(\exists x)\varphi(x)$，那么 $q\Vdash(\forall x)\neg\varphi(x)$。现在 $q\Vdash\neg(\exists x)\varphi(x)$ 说明，对于每个 $r\geqslant q$ 和 $C(r)$ 中每个 c，r 不力迫 $\varphi(c)$。根据传递性，$s\geqslant r$ 蕴涵着 $s\geqslant q$。所以对于每个 $r\geqslant q$ 和 $C(r)$ 中每个 c，没有 $s\geqslant r$ 力迫 $\varphi(c)$。这说明 $q\Vdash(\forall x)\neg\varphi(x)$。

例 2.19 如果 x 在 φ 中不是自由的，那么 $\varphi\vee(\forall x)\psi(x)\to(\forall x)(\varphi\vee\psi(x))$ 是直觉主义永真的。要证明任何 p 力迫 $\varphi\vee(\forall x)\psi(x)\to(\forall x)(\varphi\vee\psi(x))$，我们必须证明，对任何 $q\geqslant p$，如果 $q\Vdash\varphi$ 或 $q\Vdash(\forall x)\psi(x)$，那么 $q\Vdash(\forall x)(\varphi\vee\psi(x))$。存在两种情形。如果 $q\Vdash\varphi$，那么对于任何 $r\geqslant q$ 与 $C(r)$ 中任何 c，$r\Vdash\varphi\vee\psi(c)$，所以 $q\Vdash(\forall x)(\varphi\vee\psi(x))$。如果 $q\Vdash(\forall x)\psi(x)$，那么对于所有 $r\geqslant q$ 与 $C(r)$ 中所有的 c，$r\Vdash\psi(c)$，所以 $r\Vdash\varphi\vee\psi(c)$。这说明 $q\Vdash(\forall x)(\varphi\vee\psi(x))$。可将这个例子与例 2.11 相比较。

直觉主义永真性的框架定义使得我们可以很简单地证明直觉主义逻辑的两个体现其构造性的重要性质：析取性和存在性。第一个是说，如果一个析取式永真，那么它的一个析取项永真。第二个是说，如果$\mathcal{L}$的一个存在性语句永真，那么它关于$\mathcal{L}$中某个常元的一个实例也是永真的。当我们把这与直觉主义逻辑的完全性定理(定理 4.10)结合在一起时，我们看到这意味着，如果我们能证明一个存在性语句，那么我们事实上能证明某个特例。类似地，如果我们能证明一个析取式，那么我们能证明其中一个析取项。

定理 2.20(析取性) *如果$(\varphi_1\vee\varphi_2)$是直觉主义永真的，那么 φ_1 和 φ_2 之一是直觉主义永真的。*

证明 我们证明该定理的逆否命题成立。所以设 φ_1 与 φ_2 都不是直觉主义永真的。因此，对于 $i=1,2$，存在框架$\mathcal{C}_i$和偏序 R_i 的元素 p_i，使得 φ_1 在$\mathcal{C}_1$中不被 p_1 力迫且 φ_2 在$\mathcal{C}_2$中不被 p_2 力迫。根据限制引理(引理 2.4)，可设 p_i 是 R_i 中最小元素。根据习题 11，可设语言$\mathcal{L}$中没有两个不同常元在某个框架$\mathcal{C}_i(p)$的某个结构中解释为同一个元素。现在只要通过重新标记$\mathcal{C}_i(p)$(包括$\mathcal{C}_i$的所有其他结构)与 R_i 中的元素，可设$\mathcal{L}$中每个常元 c 的解释在两个结构$\mathcal{C}_i(p_i)$中是相同的且 R_i 不相交。令 R 为 R_1，R_2 和 $\{p_b\}$ 的并，其中 p_b 不在任何 R_i 中。让 R_1 和 R_2 的序保持不变，把 p_b 放到 p_1 和 p_2 的下面，得到 R 的一个偏序。



定义一个带偏序 R 的框架$\mathcal{C}$如下：对于 $p\in R_i$，令$\mathcal{C}(p)=\mathcal{C}_i(p)$，并且令$\mathcal{C}(p_b)$等同于这样的结构，其论域为$\mathcal{L}$中所有常元在结构$\mathcal{C}_i(p_i)$中的解释集(注意，对于 $i=1,2$ 这些解释是相同的)，且 $A(p_b)=\varnothing$。(我们一直假设$\mathcal{L}$至少有一个常元，这保证此结构非空。)根据限制

引理(引理 2.4)，在框架$\mathcal{C}$中 p_1 不力迫 φ_1。因此，根据单调性引理(引理 2.12)，p_b 不力迫 φ_1。类似地，由于 p_2 不力迫 φ_2，所以 p_b 不力迫 φ_2。因此，p_b 不力迫 $\varphi_1 \vee \varphi_2$；从而 $\varphi_1 \vee \varphi_2$ 不是直觉主义永真的，矛盾。 □

定理 2.21(存在性) *如果$(\exists x)\varphi(x)$是语言$\mathcal{L}$中一个直觉主义永真的语句，那么对于$\mathcal{L}$中某个常元 c，$\varphi(c)$也是直觉主义永真的。(注意，根据约定，$\mathcal{L}$中至少有一个常元。)*

证明 对于$\mathcal{L}$中每个常元 a，假设 $\varphi(a)$不是直觉主义永真的。那么，对于每个这样的常元，有一个$\mathcal{L}$框架$\mathcal{C}_a$，其偏序集 R_a 含有一个不力迫 $\varphi(a)$的元素 p_a。如同在前一个证明中那样，不失一般性，设 p_a 是 R_a 中最小的元素且所有的 R_a 互不相交。再设$\mathcal{L}$中某个固定的常元 c 解释为每个$C(p_a)$中同一个元素 d。通过取所有 R_a 的并及其偏序的并再在所有 p_a 的下面添加一个新的底元素 p_b，我们构造一个新的偏序集 R。下面定义一个带偏序 R 的$\mathcal{L}$框架。如同前一个证明中那样，令 $C(p_b)=\{d\}$，$A(p_b)=\varnothing$，且对于每个 $p\in R_a$ 和$\mathcal{L}$中每个常元 a，令 $\mathcal{C}(p)=\mathcal{C}_a(p)$。现在模仿定理 2.20 的论证。因为设$\exists x\varphi(x)$是直觉主义永真的，所以一定有 $p_b \Vdash_{\mathcal{C}} \exists x\varphi(x)$。那么根据定义，对于$\mathcal{L}$中某个常元 a，$p_b \Vdash \varphi(a)$。先应用单调性引理，再应用限制引理，我们可以得到 p_a 首先在$\mathcal{C}$中、其次在$\mathcal{C}_a$中力迫 $\varphi(a)$；这与我们最初的假设，p_a 和$\mathcal{C}_a$表明 $\varphi(a)$不是直觉主义永真的，相矛盾。 □

习题

下面的语句 1～6 是经典永真的。用框架直接验证它们是直觉主义永真的。注意，$\varphi\leftrightarrow\psi$ 是$(\varphi\rightarrow\psi)\wedge(\psi\rightarrow\varphi)$的缩写。

1. $\neg\varphi\leftrightarrow\neg\neg\neg\varphi$
2. $(\varphi\wedge\neg\psi)\rightarrow\neg(\varphi\rightarrow\psi)$
3. $(\varphi\rightarrow\psi)\rightarrow(\neg\neg\varphi\rightarrow\neg\neg\psi)$
4. $(\neg\neg(\varphi\rightarrow\psi)\leftrightarrow(\neg\neg\varphi\rightarrow\neg\neg\psi))$
5. $\neg\neg(\varphi\wedge\psi)\leftrightarrow(\neg\neg\varphi\wedge\neg\neg\psi)$
6. $\neg\neg(\forall x)\varphi(x)\rightarrow(\forall x)\neg\neg\varphi(x)$
7. 给出引理 2.4 的证明。
8. 给出引理 2.5 余下情形的证明。
9. 令 K 为出现在$(\exists x)\varphi(x)$中的常元集，且设$(\exists x)\varphi(x)$是直觉主义永真的。证明：如果 K 非空，那么对于 K 中某个 c，$\varphi(c)$是直觉主义永真的。(提示：对于语言$\mathcal{L}$的某个给定框架 $\mathcal{C}$和$\mathcal{L}$的某个限制$\mathcal{L}'$，定义$\mathcal{C}$在$\mathcal{L}'$上的限制为$\mathcal{C}'$。然后证明：对于$\mathcal{L}'$的任何语句 φ 和$\mathcal{C}'$的偏序中任何元素 p，$p\Vdash_{\mathcal{C}}\varphi$ 当且仅当 $p\Vdash_{\mathcal{C}'}\varphi$。)
10. 在前一个习题中当 K 是空集时，证明：对于每个常元 c，$\varphi(c)$是直觉主义永真的。(提示：对于$\mathcal{L}$的任何常元 a 和 c，在$\mathcal{L}$的公式上以及在$\mathcal{L}$的框架上，定义一个交换 a 与 c 的映射 Θ。证明对于任何$\mathcal{C}$和 p，$p\Vdash_{\mathcal{C}}\varphi$ 当且仅当 $p\Vdash_{\Theta(\mathcal{C})}\Theta(\varphi)$。)
11. 设$\mathcal{A}$是语言$\mathcal{L}$的一个结构，含有常元符号 c。令$\mathcal{A}'$为$\mathcal{A}$的扩张，使得 $A'=A\cup\{b_i\mid i\in\mathcal{N}\}$，其中 b_i 为不在 A 中的互不相同的元素。再把$\mathcal{L}$扩展为语言 $\mathcal{L}'$，使得$\mathcal{L}'$中有新常元 d_i 命名 b_i，并声明：对于任何 d_i 的序列$\vec{d}$和任何原子公式 ψ，$\mathcal{A}'\vDash\psi(\vec{d})$当且仅当$\mathcal{A}\vDash\psi(\vec{c})$，其中序列$\vec{c}$的每个元素都是 c。证明：对于任何 d_i 序列$\vec{d}$和$\mathcal{L}$的 任何含自由变元$\vec{x}$的公式 $\gamma(\vec{x})$，$\mathcal{A}'\vDash\gamma(\vec{d})$当且仅当$\mathcal{A}\vDash\gamma(\vec{c})$，其中 序列$\vec{c}$的每个元素都是 c。

第三节 直觉主义表

在类似于第二章第六节经典逻辑所使用的表式系统基础上，我们描述一种直觉主义逻辑的证明方法。在经典逻辑中，表证明的思想是系统地搜索一个与起始标号语句一致的结构。

我们要么得到这样的结构，要么看到每一个可能的分析都导致矛盾。当我们从一个标号语句 $F\varphi$ 开始时，要么找到一个结构在其中 φ 失败，要么断定我们有 φ 的一个证明。对于直觉主义逻辑来说，我们是从一个标号力迫断言 $Tp \Vdash \varphi$ 或 $Fp \Vdash \varphi$（φ 仍然是一个语句）开始的，并且尝试建立一个与该断言一致的框架，或者断定任何这样的尝试导致矛盾。如果从 $Fp \Vdash \varphi$ 开始，那么要么找到一个框架在其中 p 不力迫 φ，要么断定我们有 φ 的一个直觉主义的证明。

Kripke、Hughes 与 Cresswell、Fitting 等人的研究给出了很多不同的适用于直觉主义命题和谓词逻辑的表方法。我们的选择是完全针对我们的框架定义而设计的，目的是用一个系统表来表示系统地搜索一个与其起始标号力迫断言一致的框架。它由 Fitting[1983, 4.1]的前缀表变化而来。

直觉主义逻辑的表与表证明的定义在形式上非常像第二章第六节中经典逻辑的那些定义。*直觉主义表*和*表证明*是一些标签二叉树。其标签（仍然称为*表的表值*）现在是标号力迫断言，即形如 $Tp \Vdash \varphi$ 或 $Fp \Vdash \varphi$ 的标签，其中 φ 是所使用的语言中的一个语句。我们把 $Tp \Vdash \varphi$读作 p 力迫 φ，并把 $Fp \Vdash \varphi$ 读作 p 不力迫 φ。

在经典逻辑中，根据表的展开建立结构，其元素是出现在该表某条路径上的那些常元符号。我们现在来尝试建立一个完整的框架。在我们的直觉主义表中，出现在某条路径 P 上的那些 p 和 q 构成了该框架的偏序集。其偏序也是根据表的展开顺序来定义的。同经典情形一样，我们建立一个表总是要添加新常元 c_0，c_1，…，从而扩张了其起始标号断言所在的那个语言。对于 $q \leqslant p$，在路径 P 上形如 $Tq \Vdash \varphi$ 或 $Fq \Vdash \varphi$ 的表值中，出现在语句 φ 中的那些常元构成论域 $C(p)$。（我们使用 $q \leqslant p$ 的那些表值是为了保证框架定义中所要求的关于论域的单调性。）

带着这样的动机，我们定义*原子直觉主义表*（atomic intuitionistic tableau）。

定义 3.1 *我们从一个给定语言$\mathcal{L}$及其通过添加新常元符号 c_i（$i \in \mathcal{N}$）而得的扩张$\mathcal{L}_C$开始。在图 55 中列出了（语言$\mathcal{L}$的）原子直觉主义表。在这些表中，φ 和 ψ，如果没有量词，是语言$\mathcal{L}_C$中任何语句。如果有量词，它们所含的自由变元只有 x。*

在形式上，“新的 c”和“新的 p”的准确意义包含在直觉主义表的定义中。常元的用处在本质上与经典情形是相同的：当展开 $T\forall x\varphi(x)$ 时，可以用任何 c 取代 x，从而添加上 $T\varphi(c)$，但当通过添加 $T\varphi(c)$到表上而展开 $\exists x\varphi(x)$时，只能使用前面没有用过的 c。这里有一点必须注意。当我们说“任何适当的 c”时，我们指在适当的语言中的任何 c。在经典情形中这意味着$\mathcal{L}_C$中任何 c。这里，在（T∀）中像上面那样展开 $Tp \Vdash \forall x\varphi(x)$时，它的意思是指在$\mathcal{L}$中或出现在当前路径上关于某个 $q \leqslant p$ 的力迫断言中的任何 c。这些限制对应着我们想根据框架定义的要求来定义 $C(p)$，使得对于 $q \leqslant p$，有 $C(q) \subseteq C(p)$。从技术上来说，类似的考虑也可以应用于在（T∃）中使用的新的 c 上，尽管事实上可以总选取$\mathcal{L}_C$中在表任何地方都没出现的 c_i。在我们的形式定义中采用了这个选择。

在构成偏序的那些元素 p 上的限制，也应当在框架定义中加以理解。例如，在（TAt）中我们遵循框架定义的要求：如果 $p \leqslant p'$，那么 $A(p) \subseteq A(p')$。读者应当还记得，在展开表时，我们给我们的框架确定其偏序的元素，并定义该偏序自身。因此，举例来说，当展开 $Tp \Vdash \neg\varphi$ 时，根据否定式力迫的定义，对于出现在当前路径上的任何 $p' \geqslant p$，可添加 $Fp' \Vdash \varphi$。另一方面，如果我们希望加入不力迫$\neg\varphi$ 的 p，即 $Fp \Vdash \neg\varphi$，那么力迫定义说明一定有一个力迫 φ 的 $p' \geqslant p$。同添加新常元一样，我们不能使用已出现过的 p'。因此，可以像（$F\neg$）中那样展开

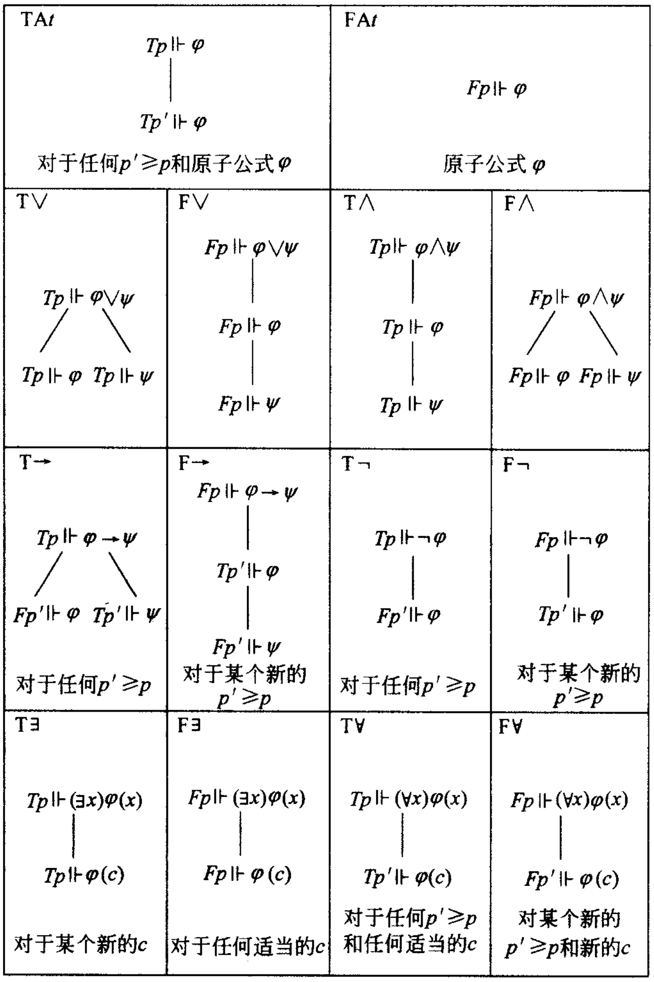


图 55

$Fp \Vdash \neg\varphi$，即对于任何新元素 p'，添加 $Tp' \Vdash \varphi$。对于这个 p'，我们只能说它在偏序中大于 p，我们要求 p'要大于 p(从而根据传递性，大于任何 $q \leqslant p$)，但 p'要与至今引入偏序的所有其他元素不可比较。(从技术上来说，我们仍然只需顾虑 p'与当前路径上出现的 q 的关系。更简单地，我们仅选用全新的 p'，即尚未在表的任何地方出现的 p'。这样的话，$p \leqslant q$ 成立仅当 p 和 q 在表的同一条路径上。)

直觉主义逻辑的表和表证明的形式定义可以留作习题。对于粗心的人来说，这可能是一个有很多陷阱的习题，所以我们详细地给出这些定义。

定义 3.2 我们继续使用给定的语言$\mathcal{L}$及其用常元所作的扩张$\mathcal{L}_C$。我们再固定一个集合 $S = \{p_i \mid i \in \mathcal{N}\}$，其中的元素作为我们力迫断言中所使用的那些 p 和 q 的潜在的候选对象。($\mathcal{L}$的)一个**直觉主义表**是一个标记着标号力迫断言的二叉树，其中这些标号力迫断言称为该表的**表值**。我们用归纳法定义由($\mathcal{L}$的)所有直觉主义表所构成的类。同时，对于每一个表 τ，我们对出现在 τ 中的 S 的元素定义一个序$\leqslant_\tau$。

(i)每个原子表 τ 是一个表。在情形(T∃)和(F∀)中，要求“c 为新的”在这里的意思

是：c 是添加到$\mathcal{L}$上得到$\mathcal{L}_C$但没在 φ 中出现的那些常元 c_i 之一。在(F∃)和(T∀)中，短语“任何 c”的意思是：$\mathcal{L}$或 φ 中任何常元。在(F→)、(F ¬)和(F∀)中，要求“p'为新的”在这里指p'是不同于 p 的任何 p_i。另外，我们声明 p'在相应的序中比 p 大。在(T→)、(T ¬)、(T∀)和(TAt)中的短语“任何 $p'\geqslant p$”在这里就意味着 p'是 p。(当然，在我们所定义的每个偏序中，我们总是声明对每一个 p 有 $p\leqslant p$。)

(ii)如果 τ 是一个有穷表，P 是 τ 上的一条路径，E 是 τ 的一个出现在 P 上的表值，且 τ'是由 τ 在路径 P 的末端添加一个根表值为 E 的原子表所得到的，那么 τ'也是一个表。对于 τ 中出现的 p_i，序$\leqslant_{\tau'}$与$\leqslant_\tau$ 是一致的。它在新元素上的行为在下面我们解释(F→)、(F ¬)和(F∀)等原子表中 对于 p'的限制的含义时给出其定义。

在情形(T∃)和(F∀)中，要求“c 是新的”在这里的意思是：它是没出现在 τ 的任何表值中的 c_i 之一(从而也不在$\mathcal{L}$中)。在(F∃)和(T∀)中，短语“任何 c”在这里指在$\mathcal{L}$中或出现在 P 上任何形如 $Tq\Vdash\psi$ 或 $Fq\Vdash\psi$ 的表值中的任何 c，其中 $q\leqslant_\tau p$。

在(F→)、(F ¬)和(F∀)中，要求“$p'\geqslant p$ 为新的”意思是我们选择没在 τ 中出现的一个 p_i 作为 p'并且声明它在$\leqslant_{\tau'}$中大于 p。(当然，为确保传递性，我们声明：对于每个 $q\leqslant_\tau p$，有 $q\leqslant_\tau p'$。)在(T→)、(T ¬)、(T∀)和(TAt)中，短语“任何 $p'\geqslant p$”的意思是：我们可选择出现在 P 上表值中且已被声明在$\leqslant_\tau$ 中大于或等于 p 的任何 p'。

(iii)如果 τ_0，τ_1，…，τ_n，…是一个有穷表的序列，使得 对于每个 $n\geqslant 0$，τ_{n+1}是 τ_n 应用一次(ii)而得，那么 $\tau=\bigcup\tau_n$ 也是一个表。

如同在谓词逻辑中那样，规定在(ii)中当添加一个原子表到 P 上时其根表值 E 要重复一次。这对于表明一个表是不是完成的是很关键的。然而，在下面的例子当中，纯粹为了符号上的方便我们一般省略这样的重复。

注意，如果我们在$\leqslant_\tau$ 的定义中不声明 $p\leqslant_\tau p'$或 $p'\leqslant_\tau p$，那么 p 和 p'在$\leqslant_\tau$ 中是不可比的。

下面的引理证实了我们前面关于序$\leqslant_\tau$ 和表 τ 中路径的关系。

引理 3.3 对于任何直觉主义表 τ 及其序$\leqslant_\tau$，如果 $p'\leqslant_\tau p$，那么 p 和 p'两个出现在 τ 的同一条路径上。

证明 根据 τ 和$\leqslant_\tau$ 的定义进行归纳证明。我们把它留作习题 31。 □

定义 3.4(直觉主义表证明) 令 τ 为一个直觉主义表且 P 是 τ 的一条路径。

(i)P 是**矛盾的**，如果对于某个力迫断言 $p\Vdash\varphi$，$Tp\Vdash\varphi$ 和 $Fp\Vdash\varphi$ 都作为一个表值出现在 P 上。

(ii)τ 是**矛盾的**，如果 τ 的每条路径是矛盾的。

(iii)τ 是 φ 的一个**直觉主义证明**(intuitionistic proof)，如果对于某个 $p\in S$，τ 是一个根节点标记着 $Fp\Vdash\varphi$ 的有穷矛盾直觉主义表。φ 是**直觉主义可证的**(intuitionistically provable)，记为$\vdash\varphi$，如果存在 φ 的一个直觉主义证明。

注意，如同在经典逻辑中那样，如果存在任一根节点为 $Fp\Vdash\varphi$ 的矛盾表，那么就存在一个矛盾的有穷表，即 φ 的证明：对每一条路径在它变矛盾时就终止它。因为这样每一条路径都是有穷的，根据库尼西引理，整棵树是有穷的。因此，要求证明必须是有穷的(表)对于任何语句证明的存在没有影响。另外一个观点是我们可以要求表定义的(ii)中路径 P 为非矛盾的，这也不会影响到证明的存在。因此，在实践中，在尝试构造一个证明时，我们用符号⊗标记出任何矛盾的路径并停止表沿着该路径展开。

在处理直觉主义逻辑表方法的可靠性和完全性之前，我们先看一些直觉主义表证明的例子。记住例子中的表省略了被展开表值的重复。我们仍然在表的左边用数字标出其层次，在右边指出该行是由哪一层的原子表展开所得。在例子中，集合 S 是一个有穷二元序列的集合，从中选取我们偏序的论域。关于序关系的声明可从每一步所添加的原子表中看出，从而也被省略。其实，我们总是选择那些 p 和 q 以使得所定义的序与二元序列上的通常的包含序相一致。

例 3.5 令 φ 和 ψ 为 $\mathcal{L}$ 的任何原子语句。图 56 提供了 $\varphi\to(\psi\to\varphi)$ 的一个直觉主义证明。

1	$F\varnothing\Vdash\varphi\to(\psi\to\varphi)$	
2	$T0\Vdash\varphi$	由1
3	$F0\Vdash\psi\to\varphi$	由1
4	$T00\Vdash\psi$	由3
5	$F00\Vdash\varphi$	由3
6	$T00\Vdash\psi$	由2
	$\otimes$	

图 56

该证明的前三行是原子表列中(F→)的一个实例。第 4 行和第 5 行是再次根据(F→)展开第 3 行而得。第 6 行根据原子表(TAt)从第 2 行得出，它与第 5 行一起得到矛盾。

例 3.6 $\mathcal{L}$ 中下列形式的任何语句是直觉主义可证的：

$$(\exists x)(\varphi(x)\vee\psi(x))\to(\exists x)\varphi(x)\vee(\exists x)\psi(x)$$

该证明(如图 57 所示)的前三行是(F→)的一个实例。第 4 行由第 2 行应用(T∃)得到。第 5 行和第 6 行由第 3 行应用(F∨)得到。第 7 行和第 8 行分别由第 5 行和第 6 行应用(F∨)得到。第 9 行由第 4 行应用(T∨)得到，它的两个分支分别与第 7 行和第 8 行矛盾。

1	$F\varnothing\Vdash(\exists x)(\varphi(x)\vee\psi(x))\to(\exists x)\varphi(x)\vee(\exists x)\psi(x)$		
2	$T0\Vdash(\exists x)(\varphi(x)\vee\psi(x))$		由1
3	$F0\Vdash(\exists x)\varphi(x)\vee(\exists x)\psi(x)$		由1
4	$T0\Vdash\varphi(c)\vee\psi(c)$		由2
5	$F0\Vdash(\exists x)\varphi(x)$		由3
6	$F0\Vdash(\exists x)\psi(x)$		由3
7	$F0\Vdash\varphi(c)$		由5
8	$F0\Vdash\psi(c)$		由6
9	$T0\Vdash\varphi(c)$	$T0\Vdash\psi(c)$	由4
	$\otimes$	$\otimes$	由7,8,9

图 57

例 3.7 考虑$(\forall x)(\varphi(x)\wedge\psi(x))\to(\forall x)\varphi(x)\wedge(\forall x)\psi(x)$。

注意这里(如图 58 所示)我们在第 4 行对两边分支同时展开，并把展开结果并列地写在

1	$F\varnothing \Vdash (\forall x)(\varphi(x)\wedge\psi(x)) \rightarrow (\forall x)\varphi(x)\wedge(\forall x)\psi(x)$		
2	$T0 \Vdash (\forall x)(\varphi(x)\wedge\psi(x))$		由1
3	$F0 \Vdash (\forall x)\varphi(x)\wedge(\forall x)\psi(x)$		由1
4	$F0 \Vdash (\forall x)\varphi(x)$	$F0 \Vdash (\forall x)\psi(x)$	由3
5	$F00 \Vdash \varphi(c)$	$F01 \Vdash \psi(d)$	由4
6	$T00 \Vdash \varphi(c)\wedge\psi(c)$	$T01 \Vdash \varphi(d)\wedge\psi(d)$	由2
7	$T00 \Vdash \varphi(c)$	$T01 \Vdash \varphi(d)$	由6
8	$T00 \Vdash \psi(c)$	$T01 \Vdash \psi(d)$	由6
	$\otimes$	$\otimes$	由5, 7 和5, 8

图 58

同一行上。还要注意的是对第 2 行应用(T∀)得到第 6 行。我们可以同时选取常元 c 和 d 以及偏序的元素 00 和 01。

习题

令 φ 和 ψ 为任何不带自由变元或在适当情形中仅有 x 自由变元的原子公式。对于下面 1 ~ 30 中每个语句 θ，构造一个由 $F\varnothing \Vdash \theta$ 开始的表以证明经典永真的 θ 也是直觉主义可证的。(记住 $\varphi\leftrightarrow\psi$ 是 $(\varphi\rightarrow\psi)\wedge(\psi\rightarrow\varphi)$ 的缩写。)

格分配律

1. $(\varphi\vee\varphi)\leftrightarrow\varphi$

2. $(\varphi\wedge\varphi)\leftrightarrow\varphi$

3. $(\varphi\wedge\psi)\leftrightarrow(\psi\wedge\varphi)$

4. $(\varphi\vee\psi)\leftrightarrow(\psi\vee\varphi)$

5. $((\varphi\wedge\psi)\wedge\sigma)\leftrightarrow(\varphi\wedge(\psi\wedge\sigma))$

6. $((\varphi\vee\psi)\vee\sigma)\leftrightarrow(\varphi\vee(\psi\vee\sigma))$

7. $(\varphi\vee(\psi\wedge\sigma))\leftrightarrow((\varphi\vee\psi)\wedge(\varphi\vee\sigma))$

8. $(\varphi\wedge(\psi\vee\sigma))\leftrightarrow((\varphi\wedge\psi)\vee(\varphi\wedge\sigma))$

纯蕴涵律

9. $\varphi\rightarrow\varphi$

10. $\varphi\rightarrow(\psi\rightarrow\varphi)$

11. $(\varphi\rightarrow(\psi\rightarrow\sigma))\rightarrow((\varphi\rightarrow\psi)\rightarrow(\varphi\rightarrow\sigma))$

$\wedge$的引入和消去

12. $(\varphi\rightarrow(\psi\rightarrow\sigma))\rightarrow((\varphi\wedge\psi)\rightarrow\sigma)$

13. $((\varphi\wedge\psi)\rightarrow\sigma)\rightarrow(\varphi\rightarrow(\psi\rightarrow\sigma))$

14. $(\neg\varphi\vee\psi)\rightarrow(\varphi\rightarrow\psi)$

德·摩根律

15. $\neg(\varphi \vee \psi) \leftrightarrow (\neg\varphi \wedge \neg\psi)$

16. $(\neg\varphi \vee \neg\psi) \rightarrow \neg(\varphi \wedge \psi)$

逆否命题

17. $(\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \neg\varphi)$

双重否定

18. $\varphi \rightarrow \neg\neg\varphi$

矛盾

19. $\neg(\varphi \wedge \neg\varphi)$

分配律

20. $(\exists x)(\varphi(x) \vee \psi(x)) \leftrightarrow (\exists x)\varphi(x) \vee (\exists x)\psi(x)$

21. $(\forall x)(\varphi(x) \wedge \psi(x)) \leftrightarrow (\forall x)\varphi(x) \wedge (\forall x)\psi(x)$

22. $(\varphi \vee (\forall x)\psi(x)) \rightarrow (\forall x)(\varphi \vee \psi(x))$，$x$ 在 φ 中不自由

23. $(\varphi \wedge (\exists x)\psi(x)) \rightarrow (\exists x)(\varphi \wedge \psi(x))$，$x$ 在 φ 中不自由

24. $(\exists x)(\varphi \rightarrow \psi(x)) \rightarrow (\varphi \rightarrow (\exists x)\psi(x))$，$x$ 在 φ 中不自由

25. $(\exists x)(\varphi \wedge \psi(x)) \rightarrow (\varphi \wedge (\exists x)\psi(x))$，$x$ 在 φ 中不自由

德·摩根律

26. $\neg(\exists x)\varphi(x) \rightarrow (\forall x)\neg\varphi(x)$

27. $(\forall x)\neg\varphi(x) \rightarrow \neg(\exists x)\varphi(x)$

28. $(\exists x)\neg\varphi(x) \rightarrow \neg(\forall x)\varphi(x)$

29. $(\exists x)(\varphi(x) \rightarrow \psi) \rightarrow ((\forall x)\varphi(x) \rightarrow \psi)$，$x$ 在 ψ 中不自由

30. $((\exists x)\varphi(x) \rightarrow \psi) \rightarrow (\forall x)(\varphi(x) \rightarrow \psi)$，$x$ 在 ψ 中不自由

31. 证明引理 3.3。

32. (**常元定理**)证明第二章第六节习题 13 的直觉主义版本：令 $\varphi(x_1, \cdots, x_n)$ 为一个公式，其中显示了其所有的自由变元，并令 $c_1, \cdots, c_n$ 为没出现在 φ 中的常元。证明存在 $\forall x_1 \cdots \forall x_n \varphi(x_1, \cdots, x_n)$ 的一个直觉主义表证明，当且仅当存在 $\varphi(c_1, \cdots, c_n)$ 的一个直觉主义表证明。

第四节 可靠性和完全性

本节第一个目标是证明在直觉主义逻辑中可证明性蕴涵着永真性。如同在经典可靠性定理(第二章定理 7.2)中那样，首先证明，一个框架如果它与一个表的根节点“一致”，那么它与该表中某个路径上的每一个表值“一致”。在经典情形中(第二章定义 7.1)，我们构造了这样的路径并定义了一个结构，其论域含该路径上标号语句中出现的常元 c。现在，除了在相应的结构 $\mathcal{C}(p)$ 中解释在路径 P 上的断言中出现的常元之外，我们必须在给定框架的序 R 中“解释”在 P 上标号力迫断言中出现的偏序元素 p。

定义 4.1 设 $\mathcal{C} = (R, \leqslant_R, \mathcal{C}(p))$ 是语言 $\mathcal{L}$ 的一个框架，τ 是一个表，其根节点标记着一个关于 $\mathcal{L}$ 的语句 φ 的力迫断言，且 P 是 τ 中一条路径。令 S 为出现在 P 上力迫断言中的 p 的集合，且令 $\leqslant_S$ 为 S 上在构造 τ 的过程中所定义的序。我们说 **$\mathcal{C}$ 与 P 一致**，如果存在映射 f 和 g，使得

(i) 如果 f 是一个从 S 到 R 的保序映射(未必是 1-1 的)。

(ii) g 把 P 上一个力迫断言 $Tp \Vdash \psi$ 或 $Fp \Vdash \psi$ 中任何语句 ψ 中出现的每一个常元 c 映射到 $\mathcal{L}(f(p))$ 中一个常元。此外，g 在 $\mathcal{L}$ 的常元上是恒等映射。我们还把 g 按照下面显然的方式扩张为公式上的映射：把 ψ 中常元 c 替换为 $g(c)$ 可得 $g(\psi)$。

(iii)如果 $Tp \Vdash \psi$ 在 P 上，那么 $f(p)$ 在 $\mathcal{C}$ 中力迫 $g(\psi)$，且如果 $Fp \Vdash \psi$ 在 P 上，那么 $f(p)$ 在 $\mathcal{C}$ 中不力迫 $g(\psi)$。

定义 4.2 设 $\mathcal{C}=(R, \leqslant_R, C(p))$ 是语言 $\mathcal{L}$ 的一个框架，且 τ 是一个表，其根部标签为关于 $\mathcal{L}$ 中语句 φ 的一个力迫断言。如果

(i) $Fr \Vdash \varphi$ 在 τ 的根部且 $q \in R$ 在 $\mathcal{C}$ 中不力迫 φ，

或者

(ii) $Tr \Vdash \varphi$ 在 τ 的根部且 $q \in R$ 在 $\mathcal{C}$ 中力迫 φ，

那么在 τ 中存在一条与 $\mathcal{C}$ 一致的路径 P；此外，存在一个证据函数 f(如定义 4.1 中所要求的那样)把 r 映射到 q。

根据表 τ 的构造，应用归纳法对这个定理进行证明。在给出其细节之前，我们把该结果重新表述为标准版本的可靠性定理。

定理 4.3(可靠性) 如果存在语句 φ 的一个直觉主义表证明，那么 φ 是直觉主义永真的。

证明(可靠性) φ 的一个直觉主义证明是一个直觉主义表 τ，其根形如 $Fr \Vdash \varphi$ 且其中每条路径都是矛盾的。如果 φ 不是直觉主义永真的，那么存在一个框架 $\mathcal{C}=(R, \leqslant_R, C(p))$ 和一个 $q \in R$，使得 q 在 $\mathcal{C}$ 中不力迫 φ。现在应用定理 4.2 得到 τ 中一条路径 P 和函数 f 与 g，满足定义 4.1 中所列的性质。因为 τ 是矛盾的，存在一个 p 和一个语句 ψ，使得 $Tp \Vdash \psi$ 和 $Fp \Vdash \psi$ 都出现在 P 上。然后由定义 4.1(iii)可得矛盾。 □

我们把定理 4.2 的归纳证明分成几个组成部分。首先，对应于定义 3.2 的(i)有 14 种情形和 14 个原子表。

引理 4.4 对于满足定理 4.2 假设的每个原子表 τ，存在其结论中所要求的 P，f 和 g。

其次，对应于定义 3.2 的(ii)中所选取展开的原子表类型有 14 种归纳情形。为了便于归纳，我们证明一个比该定理更强一些的断言。

引理 4.5 如果 f 和 g 见证某个表 τ 的路径 P 与 $\mathcal{C}$ 一致，且 τ' 由 τ 应用一次定义 3.2 的(ii)而得，那么分别存在 P，f 和 g 的扩张 P'，f' 和 g'，使得 f' 和 g' 见证 τ' 中路径 P' 也与 $\mathcal{C}$ 一致。

易证定理 4.2 是这两个引理的结果，所以我们把它的证明放在这两个引理的证明之前。

证明(定理 4.2) 引理 4.4 表明该定理对于原子表成立。然后根据归纳，用引理 4.5 证明该定理对于所有有穷表成立。事实上，它也证明该定理对于无穷表成立。设 $\tau=\cup\tau_n$ 是一个定义 3.2(iii)所定义的那样的无穷表。首先对 τ_0 应用引理 4.4 的相应情形得到适当的 P_0，f_0 和 g_0。然后依次对每个 τ_n 应用引理 4.5 构造 P_n，f_n 和 g_n。所要求的 P，f 和 g 就分别是 P_n，f_n 和 g_n 的并。 □

证明(引理 4.4) 首先定义 $f(r)=q$，并令 g 为 $\mathcal{L}$ 中常元上的恒等映射。此时，对于每个原子表的证明与引理 4.5 中对应情形的证明完全相同。这里的要点在于，对于所选取的 f 和 g，根据定理的假设可知，原子表的根节点与 $\mathcal{C}$ 一致。再对原子表的其他部分进行归纳证明可得所需要的扩展。因此，我们把该引理的证明归约为引理 4.5 的证明。 □

证明(引理 4.5) 首先注意到，如果 τ' 是由 τ 通过扩展路径 P 而得的，那么 τ 的证据也是 τ' 的证据。因此，可设 τ' 是在 τ 中 P 的末端添加一个原子表所得的。现在我们考虑定义 3.1 中原子表的 14 种情形。

情形(T∨),(F∨),(T∧),(F∧),(T→),(T¬),(F∃),(T∀)和(FAt)不需要对f或g进行扩展。显然在这些情形中，根据归纳假设和力迫定义(定义2.2)中对应情形可知，P在τ'中的扩展满足引理要求。注意(TAt)还需要假设$A(p)$上的单调性。

其他情形的证明可借用情形(F∀)的证明来描述。这里P上被展开的表值是$Fp \Vdash (\forall x)\varphi(x)$。所要求的$P$到$P'$的扩展只有一种可能，即添加$Fp' \Vdash \varphi(c)$到$P$的末端。根据我们的归纳假设，$f(p) \Vdash_{\mathcal{C}} g((\forall x)\varphi(x))$。根据力迫全称语句的定义(定义2.2(iv))，存在一个$q' \in R$和一个$c' \in \mathcal{L}(q')$，使得$q' \geq f(p)$且q'不力迫$g(\varphi(c'))$。固定这个q'和c'并通过令$f'(p') = q'$和$g'(c) = c'$将f和g扩展为f'和g'。现在显然，P'，f'和g'满足引理的要求，即，f'和g'见证P'与$\mathcal{C}$一致。

我们把情形(F→)，(F¬)和(T∃)留作习题1。 □

本节下一个目标是证明表证明法对于直觉主义逻辑是完全的。我们定义一种方法，用来构造一个适当的以给定标号力迫断言为根的完全系统表。然后证明，对于该表中任何非矛盾路径P，我们能构造一个与P一致的框架$\mathcal{C}$。因此，如果对于任何力迫断言$Fp \Vdash \varphi$应用我们的系统方法没产生φ的一个直觉主义表证明，那么我们已构造了一个框架使得φ在其中不被力迫，从而证明φ不是直觉主义永真的。

与其尝试首先抽象地定义什么时候一个表值是约化的和一个表是完成的，倒不如直接定义该构造方法，并证明它具有一些性质，这些性质可以用来证明完全性。(对于表中表值的出现，该构造方法定义"适当地展开的(properly developed)"的概念以取代"约化的"。引理4.8列出了表明一个表是完成的那些性质。)这个方法所允许的主要简化是，与其根据表的构造定义一个抽象的偏序，我们可以先定义一个特殊偏序，从中选取构造表所需要的那些p和q。任何足够丰富的偏序都可以。只要我们能从该偏序中选取一个q来扩展任何给定的p，且使得任何给定的与p不可比的元素的有穷集与q也不可比。我们选取自然数有穷序列集合S，其偏序根据序列的扩展来定义，即$p \leq q$当且仅当序列q扩展序列p。在我们表中所声明的序关系必须与S的这个序一致。我们所构造的表仅使用该序的初始片段。事实上，我们的构造是这样安排的：如果在表中序列p出现在路径P的第n层上，那么p的每个初始断都出现在P的某个$m < n$层上。

我们以原子表(T∀)和(F∀)为例，说明各种表值的处理方法。如果对于某个新的p'和c，用$Fp' \Vdash \varphi(c)$来展开$Fp \Vdash \forall x\varphi(x)$，那么我们已穷尽了包含在原断言$p$不力迫$\forall x\varphi(x)$中的信息。另一方面，对于某个$c$和$p' \geq p$，用$Tp' \Vdash \varphi(c)$展开$Tp \Vdash \forall x\varphi(x)$，则远远没有传达原断言的意思。$Tp \Vdash \forall x\varphi(x)$是说，对于(在适当的偏序中的)每个$p' \geq p$和每个$c \in \mathcal{L}(p')$，$p'$力迫$\varphi(c)$。因此，我们必须把该力迫断言所有这样的实例放在含$Tp \Vdash \forall x\varphi(x)$的每条路径上。注意，不需要用每个常元替换$\varphi$，仅使用$\mathcal{L}(p')$中的常元。同样，我们不必对于每个$p' \geq p$断言$p'$力迫$\varphi(c)$。因为我们想要构造的框架将应用完成表中非矛盾路径上的信息建立，所以我们只需考虑当前要展开的路径P上的常元。

定义4.6(完全系统直觉主义表，complete systematic intuitionistic tableau，CSIT) 令φ为语言$\mathcal{L}$的语句。令d_1，d_2，…，d_n，…列出了集合D的所有元素，其中D是语言$\mathcal{L}_C$的所有常元构成的集合，而$\mathcal{L}_C$是语言$\mathcal{L}$用新常元所得的标准扩张。为了方便，设d_1在$\mathcal{L}$中。令p_1，p_2，…，p_n，…列出了集合S的所有元素，其中S是$\mathcal{N}$中元素的所有有穷序列集，并且用序列的扩展关系确定其偏序。令v_1，v_2，…，v_k，…列出了集合V的所有元素，V是所有由D

的一个元素与 S 的一个元素所构成的有序对 $\langle d_i, p_j\rangle$ 所组成的集合。从现在起，当提到 D，S 或 V 关于某个性质的最小元时，我们指这些集合所对应的上述列表中第一个满足该性质的元素。

我们定义一个表序列 τ_n 以及 τ_n 中表值 E 的一次出现 w 是**适当地展开**的含义。表序列 τ_n 的并 τ 就是**由 φ 开始的完全系统直觉主义表**。

τ_0 是根为 $F\varnothing \Vdash \varphi$ 的原子表。如果这个表需要一个偏序元素 p' 或常元 c，那么根据定义 3.2 的(i)，选取 S 或 D 中可以用于此表的最小元。

设已构造出 τ_n。令 m 为 τ_n 中未适当展开的表值所出现在的最小层次，并令 w 为在 τ_n 的第 m 层上最左边的适当展开(比如说表值 E)的出现。我们在 τ_n 中每条含 w 的非矛盾路径 P 的末端，添加一个根为 E 的原子表构成 τ_{n+1}。为了准确地说明，我们列出 τ_n 中含 w 的所有非矛盾路径 P_1，P_2，$\cdots$，P_k。依次在 P_j 的末端添加一个根为 E 的原子表。设我们现在开始处理 P_j。情形(T∨)，(F∨)，(T∧)，(F∧)和(FAt)不再需要其他的信息来确定所添加的表。其余的每一种情形需要固定某个 p' 和(或)某个 c：

(T→)令 p' 为 S 中在 P_j 上的、扩展 p 且使得 $Fq \Vdash \varphi$ 和 $Tq \Vdash \varphi$ 都不出现在 P_j 上的最小 q。如果没有这样的 q，令 $p'=p$。

(F→)令 $k \in \mathcal{N}$ 为最小的自然数，使得 p_k 尚未出现在目前的构造中，且令 $p'=p_k$。(注意，除了 p 的初始段之外，p' 与目前已出现的任何对象都是不可比较的。)

(T¬)令 p' 为 S 中最小的 q，其中 q 在 P_j 上、扩展 p 且使得 $Fq \Vdash \varphi$ 不出现在 P_j 上。如果没有这样的 q，令 $p'=p$。

(F¬)与情形(F→)相同。

(T∃)令 c 为 D 中目前尚未出现在构造中的最小元。

(F∃)令 c 为 D 中那个在 $\mathcal{L}$ 中或出现在 P_j 上某个力迫断言 $Tq \Vdash \psi$ 或 $Fq \Vdash \psi$(其中 $q \leqslant p$)中的最小元 d，使得 $Fp \Vdash \varphi(d)$ 不出现在 P_j 上。如果没有这样的 $d \in D$，令 $c=d_1$。

(T∀)令 $\langle p', c\rangle$ 为 V 中最小的 $v=\langle r, d\rangle$，使得 r 出现在 P_j 上，d 在 $\mathcal{L}$ 中或出现在 P_j 上某个力迫断言 $Tq \Vdash \psi$ 或 $Fq \Vdash \psi$(其中 $q \leqslant p$)中，r 扩展 p 且 $Tr \Vdash \varphi(d)$ 不出现在 P_j 上。如果没有这样的有序对，令 $p'=p$ 且 $c=d_1$。

(F∀)令 $k \in \mathcal{N}$ 为最小的自然数，使得 p_k 尚未在目前的构造中出现。令 $p'=p_k$，并令 c 为 D 中尚未出现在目前的构造中的最小元。

(TAt)令 p' 为 S 中出现在 P_j 上的最小的 q，使得 $Tq \Vdash \varphi$ 不出现在 P_j 上。如果没有这样的 q，令 $p'=p$。

在所有的这些情形中我们说我们已适当地展开了表值 E 的出现 w。

在进行完全性定理的证明之前，先给出所需要的 CSIT 的一些基本性质，特别是，对应于经典的完成 CST 的那些性质。

引理 4.7 令 $\tau=\cup\tau_n$ 为如上所定义的 CSIT，且 P 是 τ 的一条路径。

(i)如果序列 $p \in S$ 出现在 P 的第 n 层上某个断言中，那么 p 的每个初始段 q 在 τ 的某个 $m \leqslant n$ 层中出现在 P 上。

(ii)根据定义 3.2，τ 是一个表。

证明 (i)根据 τ 的构造进行归纳。真正引入新 p 的情形只有(F→)和(F∀)。在这两种情形中我们对已经在 P 上的 p 引入某个序列 p_k。

(ii)仅需验证的一点是，在 τ_{n+1} 的构造中，如果在 τ_n 中某路径 P_j 的末端添加根表值为 E 的一个原子表，那么所使用的 p' 和 c(如果有的话)满足定义 3.2(ii)的条件。在其他情况下，显然遵循定义所规定的从旧表构造新表的方法。对所有情形做一个简单的检查可知我们遵守了这些规定。 □

引理 4.8 令 $\tau=\cup\tau_n$ 为如上所定义的 CSIT 且 P 是 τ 中非矛盾路径。

($T\vee$)如果 $Tp\Vdash\varphi\vee\psi$ 出现在 P 上，那么 $Tp\Vdash\varphi$ 或 $Tp\Vdash\psi$ 出现在 P 上。

($F\vee$)如果 $Fp\Vdash\varphi\vee\psi$ 出现在 P 上，那么 $Fp\Vdash\varphi$ 和 $Fp\Vdash\psi$ 出现在 P 上。

($T\wedge$)如果 $Tp\Vdash\varphi\wedge\psi$ 出现在 P 上，那么 $Tp\Vdash\varphi$ 和 $Tp\Vdash\psi$ 出现在 P 上。

($F\wedge$)如果 $Fp\Vdash\varphi\wedge\psi$ 出现在 P 上，那么 $Fp\Vdash\varphi$ 或 $Fp\Vdash\psi$ 出现在 P 上。

($T\rightarrow$)如果 $Tp\Vdash\varphi\rightarrow\psi$ 和 p' 出现在 P 上且 $p'\geqslant p$，那么 $Fp'\Vdash\varphi$ 或 $Tp'\Vdash\psi$ 出现在 P 上。

($F\rightarrow$)如果 $Fp\Vdash\varphi\rightarrow\psi$ 出现在 P 上，那么对于某个 $p'\geqslant p$，$Tp'\Vdash\varphi$ 和 $Fp'\Vdash\psi$ 出现在 P 上。

($T\neg$)如果 $Tp\Vdash\neg\varphi$ 和 p' 出现在 P 上且 $p'\geqslant p$，那么 $Fp'\Vdash\varphi$ 出现在 P 上。

($F\neg$)如果 $Fp\Vdash\neg\varphi$ 出现在 P 上，那么对于某个 $p'\geqslant p$，$Tp'\Vdash\varphi$ 出现在 P 上。

($T\exists$)如果 $Tp\Vdash\exists x\varphi(x)$ 出现在 P 上，那么对于某个 c，$Tp\Vdash\varphi(c)$ 出现在 P 上。

($F\exists$)如果 $Fp\Vdash\exists x\varphi(x)$ 出现在 P 上且 c 在 $\mathcal{L}$ 中或出现在 P 上某个力迫断言 $Tq\Vdash\psi$ 或 $Fq\Vdash\psi$(对任何 $q\leqslant p$)中，那么 $Fp\Vdash\varphi(c)$ 出现在 P 上。

($T\forall$)如果 $Tp\Vdash\forall x\varphi(x)$ 出现在 P 上，c 在 $\mathcal{L}$ 中或出现在 P 上某个力迫断言 $Tq\Vdash\psi$ 或 $Fq\Vdash\psi$(对任何 $q\leqslant p$)中，且 p' 出现在 P 上满足 $p'\geqslant p$，那么 $Tp'\Vdash\varphi(c)$ 出现在 P 上。

($F\forall$)如果 $Fp\Vdash\forall x\varphi(x)$ 出现在 P 上，那么对于某个 c 和 $p'\geqslant p$，$Fp'\Vdash\varphi(c)$ 出现在 P 上。

(TAt)如果对于任何原子 φ 和 $q\leqslant p$，p 和 $Tq\Vdash\varphi$ 出现在 P 上，那么 $Tp\Vdash\varphi$ 出现在 P 上。

证明 首先注意到，任何表值 E 在 τ 中的每个出现 w 在构造的某个阶段是适当地展开的。(考虑在 τ 的第 n 层上的任何 w。显然，如果在所有 $m\leqslant n$ 层上那些曾经被适当展开的 w' 都已经被适当地展开之后的第一个阶段 s，我们将会把 w 适当地展开。)

现在几乎立即可得情形($T\vee$)，($F\vee$)，($T\wedge$)，($F\wedge$)，($F\rightarrow$)，($F\neg$)，($T\exists$)和($F\forall$)。令 w 为适当的标号力迫条件在 P 上的出现。设我们在构造阶段 n 适当地展开了 w。因为 w 在(非矛盾的)P 上，我们在阶段 n 所处理的 P_j 之一是 P 的一个初始段。作为构造 τ_n 的一部分，在 P_j 的末端添加了适当的原子表。因此，路径 P 必定经过这个原子表的某一条分支。这立即得出所需要的结论。

接着，注意到，出现在 P 上的每个表值 E 在 P 上出现了无穷多次。这里的要点是 P 上表值 E 的每一个出现都是适当地展开的。在适当地展开 w 时，我们在 τ_n 的每条穿越 w 的非矛盾路径的末端添加 E 的另外一个出现。因此，我们确保在 P 上 E 有另外一个出现。由于 P 上每个表值 E 的每个出现都是适当地展开的，所以表值本身被无限次地适当地展开。现在容易处理引理的其余情形，其中我们只考虑以下例子。

($T\rightarrow$)为了得出矛盾，设 p' 是 p 的出现在 P 上的(在关于 S 的主列表中)最小扩展，使得 $Fp'\Vdash\varphi$ 和 $Tp'\Vdash\psi$ 都不出现在 P 上。令 Q 为关于 S 的主列表中在 p 之前的那些 $q\geqslant p$ 所构成的有穷集。根据我们对于 p' 的选择，对每个 $q\in Q$，$Fq\Vdash\varphi$ 或 $Tq\Vdash\psi$ 出现在 P 上。令 m 为 τ 构造的某个阶段，使得对每个 $q\in Q$，$Fq\Vdash\varphi$ 或 $Tq\Vdash\psi$ 出现在当前所构造的 P 的初始段中。考虑在阶段 m 之后我们首次适当地展开 $E=Tp\Vdash\varphi\rightarrow\psi$ 在 P 上某个出现的阶段 $n\geqslant m$。(因为在我们的构造中无数次地适当展开 E，所以一定有这样的阶段。)根据 CSIT 的定义，在 P 的某

个初始段 P_j 的末端添加了根为 E 且使用所给的 p'的那个原子表。因为 P_j 为 P 的初始段，这个原子表的某个分支必定是 P 的初始段，与假设矛盾。

$(T\forall)$为了得出矛盾，设 $v=\langle p',c\rangle$是(关于 V 的主列表中)满足$(T\forall)$的假设但不满足其结论的最小有序对，即 $Tp'\Vdash\varphi(c)$不出现在 P 上。令 Q 为在 v 之前并满足$(T\forall)$的假设的有序对所构成的有穷集。令 m 为某个阶段，使得对每个$\langle q,d\rangle\in Q$，$Tq\Vdash\varphi(d)$在当前所定义的 P 的初始段上已经出现了一次。考虑在 m 之后的某个阶段 n，设在这个阶段我们首次适当地展开了 $E=Tq\Vdash\forall x\varphi(x)$在 P 上的一个出现。根据 CSIT 的定义，在 P 的某个初始段 P_j 的末端添加了根为 E 且使用所给的 p'和 c 的那个原子表。因为 P_j 为 P 的初始段，这个原子表的唯一分支必定是 P 的初始段，与假设矛盾。

其他所有情形，$(T\neg)$，$(F\exists)$和(TAt)，可类似地证明。留作习题 3。 □

定理 4.9 *设 $\tau=\cup\tau_n$ 为一个 CSIT 且 P 是 τ 中非矛盾路径。我们定义 P 所对应的框架 $\mathcal{C}=(R,\leqslant,C(p))$如下：*

1. R 是 S 中出现在 P 上力迫断言中的所有序列的集合。R 上的偏序与 S 上的偏序相同——扩展。

2. 对于每个 $p\in R$，$C(p)$是$\mathcal{L}$中和出现在 P 上力迫断言 $Tq\Vdash\psi$ 或 $Fq\Vdash\psi(q\leqslant p)$中的常元所构成的集合。

3. 对于每个 $p\in R$，$A(p)$是所有 ψ 所构成的集合，其中 ψ 是原子语句，使得对于某个 $q\leqslant p$，$Tq\Vdash\psi$ 出现在 P 上。(提醒：我们采用这样的约定，即每个 $c\in C(p)$在$\mathcal{L}(p)$中用它自己命名。)

如果令 f 和 g 分别为 R 和 P 上的常元所组成的集合上的恒等映射，那么它们见证$\mathcal{C}$与 P 一致。

证明 首先注意到，根据定义 2.1，$\mathcal{C}$的定义保证$\mathcal{C}$是$\mathcal{L}$的框架。记住$\mathcal{L}(p)$中每个常元 c 用它自己命名。

我们现在希望证明$\mathcal{C}$与 P 一致；对出现在 P 上力迫断言中的语句 φ 的复杂性应用归纳法。

原子 φ：如果 $Tp\Vdash\varphi$ 出现在 P 上，那么 φ 在 $A(p)$中，从而被 p 力迫。如果 $Fp\Vdash\varphi$ 出现在 P 上，那么我们必须证明对于任何 $q\leqslant p$，$Tq\Vdash\varphi$ 不出现在 P 上。(这是 p 在$\mathcal{C}$中力迫 φ 的唯一途径。)如果在 P 上有这样一个 $Tq\Vdash\varphi$ 的出现，那么根据引理 4.8(TAt)，$Tp\Vdash\varphi$ 也会出现在 P 上，与假设 P 是不矛盾路径相矛盾。

根据引理 4.8 和力迫定义(定义 2.2)的各个条款以及关于此定理的归纳假设，即复杂性较低的语句都满足使得$\mathcal{C}$与 P 一致的那些要求，我们逐个处理对应的归纳情形。

作为例子，我们考虑 $F\forall$：$Fp\Vdash\forall x\varphi(x)$出现在 P 上。根据引理 4.8$(F\forall)$，对于某个 c 和 $p'\geqslant p$，$Fp'\Vdash\varphi(c)$出现在 P 上。然后根据归纳假设知，p'在$\mathcal{C}$中不力迫 $\varphi(c)$。再根据全称语句力迫定义(定义 2.2(v))，p 在$\mathcal{C}$中不力迫 $\forall x\varphi(x)$，结果得证。

其余情形的证明留作习题 4。 □

现在把完全性定理的标准形式表述如下。

定理 4.10 *如果 φ 是直觉主义永真的，那么它有一个直觉主义表证明。*

证明 考虑由直觉主义永真的 φ 开始的 CSIT。如果 τ 不是 φ 的直觉主义表证明，那么根据定义，它有一个不矛盾路径 P。定理 4.9 提供了一个框架$\mathcal{C}$，使得 φ 在其中不被$\varnothing$力迫。

因此 φ 不是直觉主义永真的，矛盾。 □

习题

1. 完成引理 4.5 证明的其余情形：(F→)，(F ¬)和(T∃)。
2. 证明：任何一个 CSIT 是有穷的当且仅当它是矛盾的。
3. 完成引理 4.8 证明的其余情形：(T ¬)，(F∃)和(TAt)。
4. 完成定理 4.9 证明的其余情形。

第五节 可判定性和不可判定性

CSIT 给我们提供了搜索给定语句 φ 的直觉主义证明或其框架反例的系统方法。正如我们前面所看到的那样，如果存在一个证明，那么就存在一个有穷的证明，并且根据完全性定理，CSIT 将给出这样的证明。另一方面，如果 φ 不是直觉主义永真的，完全性定理证明中所构造的框架反例通常是无穷的。事实上，一般不存在避免这种情况的方法。同经典逻辑一样，直觉主义逻辑是不可判定的：没有算法可以保证在有限时间内停止并告诉我们是否 φ 是直觉主义永真的(定理 5.16)。尽管如此，有一些特殊类型的语句，其直觉主义永真性是可以判断出来的，并且在很多情况下有一些方法可以提高我们发现证明和有穷反例的机会。

在快速高效地生成证明这个方面，完全性定理保证我们可以添加上任何生成表的可靠方法而不改变可证明语句类。也就是说，不管我们对这些生成表证明的方法作什么样的扩展，如果仍然能够证明一个可靠性定理，那么新方法(如同我们当前的方法)所生成的恰好是直觉主义永真语句的证明。当然，这里存在一个平衡。我们添加的规则越多，展开表的可能性越大。把表展开视为在所有可能的表证明中进行的搜索，那么添加新方法增加了搜索空间的宽度，但有可能降低搜索的深度，即我们所构造证明的大小。我们介绍这样一个规则，它可以大大缩短许多证明，并且允许我们调整前面所给表规则中的某种失衡现象。

为了说明附加这个规则的动机，我们首先重新考虑(如图 59 所示)例 3.5，该例对于原子语句 φ 和 ψ，证明 $\varphi\to(\psi\to\varphi)$ 是直觉主义永真的。

1	$F\varnothing\Vdash\varphi\to(\psi\to\varphi)$	
2	$T0\Vdash\varphi$	由1
3	$F0\Vdash\psi\to\varphi$	由1
4	$T00\Vdash\psi$	由3
5	$F00\Vdash\varphi$	由3
6	$T00\Vdash\varphi$	由2
	$\otimes$	由5,6

图 59

我们感兴趣的是证明的最后一步：由第 2 行根据原子表(TAt)推出第 6 行(它与第 5 行给出我们所需要的矛盾)。仅仅在证明的这个地方，即在应用原子表 TAt 时，我们才用到 φ 和 ψ 是原子语句的事实。我们所选择的这个规则对应于定义一种框架，它们仅对原子表要求单调性。此外，单调性引理(引理 2.12)表明，该规则对于任何语句 φ 是可靠的：如果 $p\Vdash\varphi$ 且 $p'\geqslant p$，那么在任何框架中 $p'\Vdash\varphi$。因此可以修改我们的表方法，使得原子表 TAt 可应用于所有语句 φ：

$$\boxed{\begin{array}{l} T\varphi \\ \quad Tp\Vdash\varphi \\ \quad\quad | \\ \quad Tp'\Vdash\varphi \\ \text{对于任何}p'\geqslant p \end{array}}$$

单调性

对于这个加强的原子表与 φ 为原子语句的原来那个表来说，可靠性定理的证明是完全相同的。因此我们并没有改变可证明语句类，但是却大大提高了我们证明的适用性。现在看来，在第三节中假设 φ 和 ψ 为原子语句的那些例子和习题中，表证明都可用于证明对于任何语句 φ 和 ψ，给定表达式的永真性。事实上，由于(TAt)是唯一的不能普遍地应用于所有语句的表展开方式，我们现在看到由原子 φ，ψ 等所构成的某个语句的任何表证明，其实证明了该语句用任意公式替换 φ，ψ 等所得的所有实例。(提醒：这些替换必须遵守一个规定。因为我们的表方法仅适用于语句，我们必须当心不要在我们的公式中引入任何自由变元。)

现在再应用该表证明法对于非直觉主义永真的语句构造其框架反例。注意，这里没有肯定有结果的方法。然而我们可以把完全性定理的证明转变成更一般的、对是否已构造出一个框架反例的验证。这需要在引理 4.8 的基础上给出完成表的概念。

定义 5.1 *令 τ 是一个表且 $\leqslant$ 是定义在 τ 中所出现的那些 p 和 q 上的偏序。如果 τ 中每条非矛盾路径 P 具有引理 4.8 所列的那 13 个性质，那么称 τ 是*完成的*。*

定理 5.2 *如果 τ 是一个完成表，其根为 $Fp \Vdash \varphi$ 且 P 是 τ 中一条非矛盾路径，那么存在一个与 P 一致的框架 $\mathcal{C}$(从而 φ 不是直觉主义永真的)。*

证明 我们完全可以像在定理 4.9 中那样进行证明，除了在 P 的力迫断言中出现的那些 p 上的序现在是根据 τ(而不是根据事先给定的序列扩展)来定义的之外。除了引理 4.8 中所给出的性质之外，定理 4.9 的证明没用到 CSIT 的其他性质。现在根据 τ 是完成的的定义，P 满足这些性质。 □

因此，如果我们能产生一个根节点形如 $Fp \Vdash \varphi$ 的非矛盾的完成表，那么我们就构造出了 φ 的一个框架反例，从而证明它不是直觉主义永真的。

例 5.3 尝试证明非永真语句

$$\varphi \to (\varphi \to \psi)$$

首先展开其中两个蕴涵直到得出如图 60 所示的第 5 行。现在如果 φ 和 ψ 是原子语句，易知该表是完成的。因为它是非矛盾的，所以我们已证明对于原子语句 φ 和 ψ 来说，$\varphi \to (\varphi \to \psi)$ 不是直觉主义永真的。

我们可以再作进一步的分析，实际地给出一个框架，使得 $\varphi \to (\varphi \to \psi)$ 在其中不被力迫；这样的框架可根据上面图 60 所示的表构造出来。

1	$F\varnothing \Vdash \varphi \to (\varphi \to \psi)$	
2	$T0 \Vdash \varphi$	由1
3	$F0 \Vdash (\varphi \to \psi)$	由1
4	$T00 \Vdash \varphi$	由3
5	$F00 \Vdash \psi$	由3

图 60

$$\begin{array}{c} \langle\{c\},\{\varphi\}\rangle \\ | \\ \langle\{c\},\{\varphi\}\rangle \\ | \\ \langle\{c\},\varnothing\rangle \end{array}$$

当然，我们也可以删除该框架最上面的那一行，因为它对于第 2 行而言没有添加任何东西。我们还看到这个框架给我们提供了一个模板，对于非原子的 φ 和 ψ，用它可以构造出 $\varphi \to (\varphi \to \psi)$ 的反例。如果我们让 $\varnothing$ 不力迫 φ 而 0 力迫 φ 但不力迫 ψ，我们将得到所需要的反例。

例 5.4 考虑 $\varphi \vee \neg\varphi$(排中律)和如图 61 所示的表。

1	$F\varnothing \Vdash \varphi \vee \neg\varphi$	
2	$F\varnothing \Vdash \varphi$	由1
3	$F\varnothing \Vdash \neg\varphi$	由1
4	$T0 \Vdash \varphi$	由3

图 61

我们再一次看到，如果 φ 和 ψ 是原子语句，那么我们有一个完成表，从而证明 $\varphi \vee \neg\varphi$ 不是直觉主义永真的。对应于这个完成表的框架反例与例 2.7 中所产生的那个框架是一样的。这里的差别在于后者是自动产生的。

例 5.5 考虑语句 $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$ 和如图 62 所示的表。

这不是一个完成表，但如果我们足够聪明的话，我们能用它构造出一个框架反例。关键思想在于，对任何新的 θ 的任何进一步展开都不会产生真力迫断言 $Tp \Vdash \theta$。因此我们必定拥有足以构造一个反例的信息。事实上，令 $\varnothing$ 和 0 不力迫任何原子命题 $00 \Vdash \psi$，$000 \Vdash \psi$ 和 $000 \Vdash \varphi$，可得所需要的框架反例。现在，从力迫的观点来看，$\varnothing$ 与 0 是没有区别的。因此我们可以将它们合二为一。所以最后得到的偏序集为 $\{\varnothing, 0, 00\}$，且

$$A(\varnothing) = \varnothing, \quad A(0) = \{\psi\}, \quad A(00) = \{\varphi, \psi\}$$

我们把验证该框架确实为一个框架反例留作习题 13。

例 5.6 考虑语句 $(\varphi \to \psi) \vee (\psi \to \varphi)$ 和如图 63 所示的表。

1	$F\varnothing \Vdash (\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$		
2	$T0 \Vdash \neg\varphi \to \neg\psi$		由1
3	$F0 \Vdash \psi \to \varphi$		由1
4	$T00 \Vdash \psi$		由3
5	$F00 \Vdash \varphi$		由3
7	$F00 \Vdash \neg\varphi$	$T00 \Vdash \neg\psi$	由2
8	$T000 \Vdash \varphi$	$F00 \Vdash \psi$	由7
		$\otimes$	由4,8

图 62

1	$F\varnothing \Vdash (\varphi \to \psi) \vee (\psi \to \varphi)$	
2	$F\varnothing \Vdash \varphi \to \psi$	由1
3	$F\varnothing \Vdash \psi \to \varphi$	由1
4	$T0 \Vdash \varphi$	由2
5	$F0 \Vdash \psi$	由2
6	$T1 \Vdash \psi$	由3
7	$F1 \Vdash \varphi$	由3

图 63

我们看到这是第一个例子，其中规则(F→)的规定"新的"$p' \geqslant p$(应用于第 3 行得出第 6 行和第 7 行)使得我们的框架分支延伸(branch)。根据 CSIT 的定义 4.6 中的规则(F→)，第 6 行中所选取的节点 1 是大于 $\varnothing$ 且与树上不 $\leqslant \varnothing$ 的每个 p 是不可比较的最小节点。事实上，没有线性(不分叉)框架不力迫 $(\varphi \to \psi) \vee (\psi \to \varphi)$。然而上面的表是完成的且是不矛盾的，所以 $(\varphi \to \psi) \vee (\psi \to \varphi)$ 不是直觉主义永真的。

我们最后再看一个例子，其中的语句不是直觉主义永真的，且其 CSIT 在展开若干步之后不能明显地给出一个框架反例。事实上，这个例子表明不存在有穷框架反例。

例 5.7 考虑语句 $\forall x \neg\neg\varphi(x) \to \neg\neg\forall x\varphi(x)$。

这个语句的 CSIT 的开始部分如图 64 所示，其形式已作了一些简化。

不容易看出从该 CSIT 的这些初始阶段如何构造一个框架反例。直接进行语义分析可知，

1	$F\varnothing \Vdash (\forall x)\neg\neg\varphi(x) \to \neg\neg(\forall x)\varphi(x)$	
2	$T0 \Vdash (\forall x)\neg\neg\varphi(x)$	由1
3	$F0 \Vdash \neg\neg(\forall x)\varphi(x)$	由1
4	$T0 \Vdash \neg\neg\varphi(c)$	由2
5	$T00 \Vdash \neg(\forall x)\varphi(x)$	由3
6	$T00 \Vdash \neg\neg\varphi(c)$	由2
7	$F0 \Vdash \neg\varphi(c)$	由4
8	$F00 \Vdash (\forall x)\varphi(x)$	由5
9	$F00 \Vdash \neg\varphi(c)$	由6
10	$T01 \Vdash \varphi(c)$	由7
11	$F000 \Vdash \varphi(c_1)$	由8

图 64

我们应当不断地引入新常元 c_1，c_2，…，且保证当下没被力迫的 $\varphi(c_n)$ 要在后面某个扩展中被力迫。我们可以看到这一现象出现的原因在于，一方面由第 4 行生成第 7 行和第 10 行所构成的循环，另一方面由第 5 行生成第 8 行和第 11 行所构成的循环。

图 65 是一个简化的框架，它给出了一个反例。

我们把验证 $\varnothing$ 在上述框架中不力迫 $\forall x\neg\neg\varphi(x) \to \neg\neg\forall x\varphi(x)$ 留作习题 15。然而，我们要证明不存在有穷框架反例。

……

$\langle\{c,c_1,c_2,c_3\},\{\varphi(c),\varphi(c_1),\varphi(c_2)\}\rangle$

$\langle\{c,c_1,c_2\},\{\varphi(c),\varphi(c_1)\}\rangle$

$\langle\{c,c_1\},\{\varphi(c)\}\rangle$

$\langle\{c\},\varnothing\rangle$

图 65

命题 5.8 *在每个带有穷偏序 R 的框架 $\mathcal{C}$ 中，每个节点 p 力迫 $\forall x\neg\neg\varphi(x)\to\neg\neg\forall x\varphi(x)$。*

证明 令 p，$\mathcal{C}$ 和 R 如命题中所指。要验证 $p \Vdash \forall x\neg\neg\varphi(x)\to\neg\neg\forall x\varphi(x)$，考虑任何 $q\geqslant p$ 使得 $q \Vdash \forall x\neg\neg\varphi(x)$。我们必须证明 $q \Vdash \neg\neg\forall x\varphi(x)$。根据引理 2.13，这等于说，对每个 $r\geqslant q$，存在 $s\geqslant r$，使得 $s \Vdash \forall x\varphi(x)$。固定某个 $r\geqslant q$。因为 R 是有穷的，所以在 R 中存在 r 的一个极大扩展 s。现在根据单调性，$s \Vdash \forall x\neg\neg\varphi(x)$。因此，对于任何 $c\in C(s)$，$s \Vdash \neg\neg\varphi(c)$。再次应用引理 2.13，以及 s 的极大性，可得 $s \Vdash \varphi(c)$。因此(再次根据 s 的极大性)，$s \Vdash \forall x\varphi(x)$。 □

在例 5.7 中，没有有穷框架反例的那些语句都含有量词，这并非偶然。只有含量词语句

的反例才一定是无穷框架。任何不含量词的语句，要么是直觉主义永真的，要么有一个有穷框架反例。这个事实给我们提供了一个关于无量词公式直觉主义永真性的判定方法(尽管简单)。

定理 5.9(有穷模型性质)　一个无量词语句在所有框架中被力迫，当且仅当它在所有有穷框架中被力迫。

证明　考虑任何无量词公式 φ。我们必须证明：如果它在某个框架$\mathcal{C}$中不被某个 $p\in R$ 所力迫，那么存在一个有穷框架$\mathcal{C}'$，使得它在其中不被力迫。令 X 为 φ 的所有子公式集。(回忆第二章判断 φ 的子公式的定义 2.6 或命题 3.8。)对于 R 中的 p，定义 R 的元素所组成的类$[p]$，其中每个元素与 p 力迫 X 中相同的公式：

$$[p] = \{q \in R \mid (\forall \psi \in X)(p \Vdash \psi \leftrightarrow q \Vdash \psi)\}$$

对于$[p]\in R$，令 R'为所有这样的$[p]$所组成的集合。现在不同类$[p]\in R'$对应着 X 的不同子集。因为 X，即 φ 的子公式集，是有穷的，所以 R'也是有穷的。定义 R'上的偏序为：$[q]\leqslant[p]$，如果 X 中被 q 力迫的每个公式都被 p 力迫。(根据$[p]$和$[q]$的定义，这等同于要求 X 中被$[q]$中某个 r 所力迫的每个公式也都被$[p]$中某个 s 所力迫。)定义一个以 R'为其偏序集的有穷框架$\mathcal{C}'$，使得 $A([p]) = A(p)\cap X$，且令 $C'([p])$为出现在 $A([p])$中的常元集。我们断言：对于所有 $p\in R$ 和 X 中的所有 ψ，$[p]\Vdash_{\mathcal{C}'}\psi$ 当且仅当 $p\Vdash_{\mathcal{C}}\psi$。这个断言显然足以证明该定理。我们对公式进行归纳来证明该断言：

原子 ψ：$A([p]) = A(p)\cap X$ 是说，如果 ψ 是 X 中的原子公式，那么$[p]\Vdash_{\mathcal{C}'}\psi$，当且仅当 ψ 在 $A(p)\cap X$ 中，或等价地，当且仅当 $p\Vdash\psi$。

归纳步：设 θ 和 ψ 在 X 中。根据归纳，设对所有 $q\in R$，$q\Vdash_{\mathcal{C}}\theta$ 当且仅当$[q]\Vdash_{\mathcal{C}'}\theta$，且 $q\Vdash_{\mathcal{C}}\psi$当且仅当$[q]\Vdash_{\mathcal{C}'}\psi$。

(1)析取式：$p\Vdash_{\mathcal{C}}\theta\vee\psi\Leftrightarrow p\Vdash_{\mathcal{C}}\theta$ 或 $p\Vdash_{\mathcal{C}}\psi\Leftrightarrow[p]\Vdash_{\mathcal{C}'}\theta$ 或$[p]\Vdash_{\mathcal{C}'}\psi$(根据归纳)$\Leftrightarrow[p]\Vdash_{\mathcal{C}'}\theta\vee\psi$。

(2)合取式：$p\Vdash_{\mathcal{C}}\theta\wedge\psi\Leftrightarrow p\Vdash_{\mathcal{C}}\theta$ 且 $p\Vdash_{\mathcal{C}}\psi\Leftrightarrow[p]\Vdash_{\mathcal{C}'}\theta$ 且$[p]\Vdash_{\mathcal{C}'}\psi$(根据归纳)$\Leftrightarrow[p]\Vdash_{\mathcal{C}'}\theta\wedge\psi$。

(3)蕴涵式：设$[p]\Vdash_{\mathcal{C}'}\theta\rightarrow\psi$。我们必须证明 $p\Vdash_{\mathcal{C}}\theta\rightarrow\psi$。如果 $q\geqslant p$ 且 $q\Vdash_{\mathcal{C}}\theta$，那么根据归纳，$[q]\Vdash_{\mathcal{C}'}\theta$，所以根据我们的假设和由 $q\geqslant p$ 可得$[q]\geqslant[p]$的事实，$[q]\Vdash_{\mathcal{C}'}\psi$。然后由归纳假设，可得所要求的 $q\Vdash_{\mathcal{C}}\psi$。反之，设 $p\Vdash_{\mathcal{C}}\theta\rightarrow\psi$ 且 $\theta\rightarrow\psi$ 在 X 中。我们必须证明$[p]\Vdash_{\mathcal{C}'}\theta\rightarrow\psi$，即如果$[q]\geqslant[p]$且$[q]\Vdash_{\mathcal{C}'}\theta$，那么$[q]\Vdash_{\mathcal{C}'}\psi$。现在因为根据假设，$\theta\rightarrow\psi$ 在 X 中且 $p\Vdash_{\mathcal{C}}\theta\rightarrow\psi$，所以$[q]\geqslant[p]$蕴涵 $q\Vdash_{\mathcal{C}}\theta\rightarrow\psi$。根据我们的假设$[q]\Vdash_{\mathcal{C}'}\theta$ 和归纳假设，有 $q\Vdash_{\mathcal{C}}\theta$。因此$q\Vdash_{\mathcal{C}}\psi$，且再次根据归纳，有$[q]\Vdash_{\mathcal{C}'}\psi$。

(4)否定式的证明与蕴涵式的类似，留作习题 16。 □

定理 5.10　我们可以能行地判断任何无量词语句的直觉主义永真性。

证明　根据完全性定理(定理 4.9 和定理 4.10)中所表达的 CSIT 的性质，我们知道如果一个给定语句 φ 是直觉主义永真的，那么其 CSIT 将给出 φ 的一个(必然)有穷的表证明。另一方面，如果 φ 不是永真的，那么根据有穷模型性质(定理 5.9)，存在一个有穷框架反例。因此我们可以同时搜索 φ 的有穷框架反例和展开 φ 的 CSIT。最终必定找到 φ 的一个有穷框架反例，或者得到 φ 是直觉主义永真的直觉主义表证明。 □

定理 5.10 的证明中所体现的判断方法不是很令人满意。根据该方法，我们既不知道其证明或反例到底有多大，也不知道我们要搜索多久才能找到。我们所能说的仅仅是，给定一个无量词语句 φ，只需考虑出现在所给语句中的原子公式的所有框架，它们至多有 φ 的所有

子公式那么多。但是通过进一步努力，我们可以有一个更明确的算法，该算法可以产生无量词的 φ 的某个表证明，而且产生一个证明或反例所需的步数可以控制在更好的界限内（见 Nerode[1990, 4.2]）。我们只需指出的是，该判断方法比经典逻辑所用的判断方法要复杂得多。正如我们所知道的（第一章定理 4.8），无量词的谓词逻辑等同于命题逻辑。经典命题逻辑中可满足性的判定问题是典型的 NP-完全问题。但是对于命题直觉主义逻辑来说，该判定问题对多项式空间是完全的（Statman[1979, 5.3]）。

直觉主义逻辑中的可判定性可以再向前推进一步。与经典逻辑不同，前束语句类的直觉主义永真性是可判定的。上面（或其他的）关于无量词语句的证明可以扩展到前束语句（习题 17），需要应用存在性（定理 2.21）和常元定理的一个直觉主义版本（第三节习题 32）。然而，正如人们所预料的那样，整个直觉主义逻辑的永真性问题，同经典逻辑一样，是不可判定的。给定经典逻辑永真性问题的不可判定性（第三章推论 7.10），我们可以用哥德尔给出的保持永真性的变换把它归约为直觉主义逻辑永真性问题的不可判定性。

定义 5.11 如果 A 是一个原子公式，那么 $\neg\neg A$ 是**哥德尔公式**。如果 φ 和 ψ 是哥德尔公式，那么 $\neg\varphi$，$\varphi\wedge\psi$ 和 $\forall x\varphi$ 也是。

在经典逻辑中，A 和 $\neg\neg A$ 是等价的且 $\{\neg, \wedge, \forall\}$ 是充分的（adequate）。所以对于每个公式 φ，存在一个歌德尔公式 φ° 等价于 φ。经典永真性的判定问题因此可以约化为仅仅判定哥德尔公式的永真性。我们现在希望证明：一个哥德尔公式 ψ 是经典永真的，当且仅当它是直觉主义永真的。其中充分性方向就是定理 2.6 的特例。我们需要对哥德尔公式证明其逆命题。

定义 5.12 如果 φ 是一个哥德尔公式且 p 是框架 $\mathcal{C}$ 中的一个力迫条件，那么 $p\Vdash\varphi$ 或 $(\exists q\geq p)(q\Vdash\neg\varphi)$。特别地，如果 p 不力迫 φ，那么下列情形之一成立：

(1) 如果 $\varphi=\neg\psi$，那么 $\exists q\geq p(q\Vdash\psi)$。

(2) 如果 $\varphi=\psi\wedge\theta$，那么 $\exists q\geq p(q\Vdash\neg\psi$ 或 $q\Vdash\neg\theta)$。

(3) 如果 $\varphi=\forall x\psi$，那么 $(\exists q\geq p)(\exists c\in C(q))(q\Vdash\neg\psi(c))$。

证明 我们对 φ 用归纳法。其基本情形是 φ 为 $\neg\neg A$，其中 A 为某个原子语句。在这种情形中，如果 p 不力迫 φ，那么根据否定式力迫的定义（定义 2.2(iii)），存在 $q\geq p$ 力迫 $\neg A$，这就是(1)所要求的。注意，一般来说，如果 $\varphi=\neg\psi$ 且 $q\Vdash\psi$，那么根据 $\psi\to\neg\neg\psi$ 的直觉主义永真性（例 2.15），有 $q\Vdash\neg\neg\psi$（即 $q\Vdash\neg\varphi$）。

如果 φ 为 $\neg\psi$ 且 p 不力迫 $\neg\psi$，那么，同基本情形一样，存在 $q\geq p$ 力迫 ψ 以及 $\neg\neg\psi$。

如果 φ 为 $\psi\wedge\theta$ 且 p 不力迫 φ，则要么 p 不力迫 ψ，要么 p 不力迫 θ。因此根据归纳，存在 $q\geq p$ 力迫 $\neg\psi$ 或 $\neg\theta$。（再一次注意，根据关于力迫的基本事实，这蕴涵着 $q\Vdash\neg(\psi\wedge\theta)$。）

如果 φ 为 $\forall x\psi(x)$ 且 p 不力迫 φ，那么根据力迫定义，存在 $r\geq p$ 和 $c\in C(r)$，使得 r 不力迫 $\psi(c)$。那么由归纳假设知，存在 $q\geq r$，使得 $q\Vdash\neg\psi(c)$，这正是所要求的。当然任何这样的 q 也力迫 $\neg\varphi$。 □

定义 5.13 框架 $\mathcal{C}$ 中一个力迫条件序列 $\langle p_i\rangle$ 是**扩展 p 的一般序列**（generic sequence extending p），如果 $p_0=p$ 且下列条件成立：

(i) 对每个 i，$p_i\leq p_{i+1}$。

(ii) 对每个原子语句 ψ，存在 i，使得 p_i 力迫 ψ 或 p_i 力迫 $\neg\psi$。

(iii) 对每个哥德尔语句 φ，存在 i，使得 $p_i\Vdash\varphi$ 或者 p_{i+1} 是引理 5.12 中相应条款所要求

的关于 φ 的某个力迫条件 $q \geqslant p_i$。

引理 5.14 对于框架$\mathcal{C}$中每个力迫条件 p，存在一个扩展 p 的一般序列。

证明 令$\{\varphi_i \mid i \in \mathcal{N}\}$为所有哥德尔语句的一个枚举。归纳定义一个序列$\langle p_i \mid i \in \mathcal{N}\rangle$。取 $p_0 = p$。如果 $p_i \Vdash \varphi_i$ 且 φ_i 为$\neg\neg\psi$，其中 ψ 为原子语句，那么根据力迫定义，存在 $q \geqslant p_i$ 力迫 ψ。令 p_{i+1}为这样的 q。如果 $p_i \Vdash \varphi_i$ 但 φ_i 不是这种形式的，令 $p_{i+1} = p_i$。如果 p_i 不力迫 φ_i，令 p_{i+1}扩展 p_i 的一个条件，这可由引理 5.12 中对应于 φ_i 的条款所保证。(所以，特别地，如果 φ_i 是$\neg\neg\psi$，其中 ψ 为原子语句且 p_i 不力迫 φ_i，那么 $p_{i+1} \Vdash \neg\psi$。)显然，序列 p_i 满足扩展 p 的一般序列的定义。 □

定理 5.15 对每个哥德尔语句 φ，φ 是经典永真的，当且仅当 φ 是直觉主义永真的。

证明 正如我们所说的那样，如果 φ 是直觉主义永真的，那么它是经典永真的(定理 2.6)。要证明其逆命题，设 φ 不是直觉主义永真的，即存在一个框架$\mathcal{C}$和一个力迫条件 p，使得 p 不力迫 φ。我们必须建立一个经典模型$\mathcal{A}$，使得 φ 在其中为假。根据引理 5.14，我们可以选取哥德尔语句的一个枚举，其中 $\varphi_0 = \varphi$，和$\mathcal{C}$中一个扩展 p 的一般序列$\langle p_i \mid i \in \mathcal{N}\rangle$。注意，根据我们的假设 p 不力迫 φ 和一般序列的定义，有 $p_1 \Vdash \neg\varphi$。令我们所要求的经典模型$\mathcal{A}$的全域 A 为$\bigcup\{C(p_i) \mid i \in \mathcal{N}\}$。我们定义$\mathcal{A}$的关系为$\mathcal{A} \vDash R(\vec{c})$当且仅当$\exists i(p_i \Vdash R(\vec{c}))$。

我们断言：对于每个哥德尔语句 φ，$\mathcal{A} \vDash \varphi \Leftrightarrow \exists i(p_i \Vdash \varphi)$。因为 $p_1 \Vdash \neg\varphi$(且$\neg\varphi$ 是哥德尔语句)，这给了我们所要求的$\neg\varphi$ 的经典模型。其证明可根据语句 φ 的构造进行归纳。(注意，我们在定义 5.11 中指定了公式构造层次的顺序，使得对于每一个常元 c，$\forall x\psi(x)$都在$\neg\psi(c)$的后面。)

基本情形是 φ 为$\neg\neg\psi$，其中 ψ 为原子语句。因为 p_i 是一个一般序列，所以存在一个 i，使得 p_i 力迫 ψ 或 p 力迫$\neg\psi$。在前一种情形中，根据定义，$\mathcal{A}$满足 ψ(以及$\neg\neg\psi$)。在后一种情形中，没有 p_j 能够力迫 ψ。所以根据$\mathcal{A}$的定义，ψ 在$\mathcal{A}$中为假。

如果 φ 为 $\psi \wedge \theta$，那么$\mathcal{A} \vDash \varphi \Leftrightarrow \mathcal{A} \vDash \psi$ 且$\mathcal{A} \vDash \theta$。根据归纳，这个条件成立当且仅当存在 j 和 k，使得 $p_j \Vdash \psi$ 且 $p_k \Vdash \theta$。因为 p_i 构成力迫条件的递增序列，所以这等价于存在 i，使得 $p_i \Vdash \psi \wedge \theta$，即 $p_i \Vdash \varphi$。

如果 φ 是$\neg\psi$，那么$\mathcal{A} \vDash \varphi \Leftrightarrow \mathcal{A} \nvDash \psi$。根据归纳，这成立当且仅当不存在 j，使得 $p_j \Vdash \psi$。根据一般序列定义，这最后一个条件等价于存在 i，使得 $p_i \Vdash \neg\psi$，即 $p_i \Vdash \varphi$。

如果 φ 为$\forall x\psi(x)$，先设$\mathcal{A} \vDash \varphi$。如果不存在 p_i 力迫 φ，那么根据一般序列定义，存在 i 和 c，使得 $p_i \Vdash \neg\psi(c)$。那么根据归纳，$\mathcal{A} \vDash \neg\psi(c)$，矛盾。对于逆命题，设存在 i，使得 $p_i \Vdash \forall x\psi(x)$。那么，对每个 $c \in A$，存在 $j \geqslant i$，使得 $c \in C(p_j)$且 $p_j \Vdash \psi(c)$。从而根据归纳，我们知道$\mathcal{A} \vDash \psi(c)$对于每个 $c \in A$ 成立，即$\mathcal{A} \vDash \varphi$，得证。 □

定理 5.16 直觉主义逻辑的永真性问题是不可判定的。

证明 如果我们可以能行地判定任意给定的语句 ψ 是不是直觉主义永真的，那么我们就可以通过检查(如同定义 5.11 中所定义的)φ°是不是直觉主义永真的来判定任何语句 φ 是不是经典永真的。这与经典谓词逻辑永真性的不可判定性(第三章推论 7.10)矛盾。 □

推论 5.17 不是每一个语句都直觉主义等价于一个前束形式语句。

证明 假如每个语句都有一个直觉主义等价的前束形式，系统的搜索将找到这样一个等价的表证明。那么前束语句永真性的判定方法(习题 17)可用于判定所有语句的永真性。 □

习题

下面是一列(1～12)经典永真的但不是直觉主义永真的语句 θ。对其中的每一个语句，从 $F\varnothing \Vdash \theta$ 开始展开表直到足以产生一个框架，使得 θ 在其中不被力迫。在每一个例子中，设 φ 和 ψ 为原子语句且要么不含自由变元(1～8)，要么只有 x 是自由的(9～14)。

1. $(\varphi \vee \neg\varphi)$

2. $(\neg\neg\varphi \to \varphi)$

3. $\neg(\varphi \wedge \psi) \to (\neg\varphi \vee \neg\psi)$

4. $\neg\varphi \vee \neg\neg\varphi$

5. $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$

6. $(\varphi \to \psi) \to (\neg\varphi \vee \psi)$

7. $\neg(\forall x)\varphi(x) \to (\exists x)\neg\varphi(x)$

8. $(\forall x)\neg\neg\varphi(x) \to \neg\neg(\forall x)\varphi(x)$

9. $(\forall x)(\varphi \vee \psi(x)) \to (\varphi \vee (\forall x)\psi(x))$

10. $(\varphi \to (\exists x)\psi(x)) \to (\exists x)(\varphi \to \psi(x))$

11. $((\forall x)\varphi(x) \to \psi) \to (\exists x)(\varphi(x) \to \psi(x))$

12. $((\forall x)(\varphi(x) \vee \neg\varphi(x)) \wedge \neg\neg(\exists x)\varphi(x)) \to (\exists x)\varphi(x)$

13. 证明：在例 5.5 所构造的框架中 $\varnothing$ 不力迫 $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$。

14. 证明：$(\varphi \to \psi) \to (\psi \to \varphi)$ 在每个框架 $\mathcal{C}$ 中被每个节点所力迫，其中 $\mathcal{C}$ 的偏序集 R 上的序是线性序。

15. 证明：在例 5.7 所构造的框架中 $\varnothing$ 不力迫 $\forall x\,\neg\neg\varphi(x) \to \neg\neg\forall x\varphi(x)$。

16. 完成定理 5.9 的证明，即验证归纳步中的否定式情形。

17. 证明：前束语句类的直觉主义永真性是可判定的。(提示：连续应用第二节习题 9 和第三节习题 32，将判定前束形式语句的永真性归约为判定许多无量词语句的永真性。)

第六节 比较指南

本节提供了一个比较模态逻辑和直觉主义逻辑之间异同的指南。我们以第二章中所给出的经典谓词逻辑作为一个共同的起点。

一、语法

第二章定义 2.1 到定义 2.6 中给出了经典逻辑的语法。由于技术上的原因，在模态逻辑和直觉主义逻辑中，设语言 $\mathcal{L}$ 至少含有一个常元符号，但不含任何不同于常元的函数符号。我们仍然把 $\leftrightarrow$ 视为一个被定义的符号

$$\varphi \leftrightarrow \psi \quad \text{意思是} \quad \varphi \to \psi \wedge \psi \to \varphi$$

当然，模态逻辑添加了两个新算子 $\Box$ 和 $\Diamond$，在语法上可视为一元的命题联结词。然而它们的语义实际上是二阶的。

二、语义

第二章第四节中给出了经典逻辑的语义，即在语言 $\mathcal{L}$ 的结构 $\mathcal{A}$ 中解释 $\mathcal{L}$。模态逻辑和直觉主义逻辑两者使用结构系统(称为框架)和一个新关系(力迫，$\Vdash$)来定义它们的语义。在两种情形中，框架的构成包括一个可能世界集(W 或 R)、这个集合上的二元关系 S 和一个函数，该函数给每个世界 p 指派一个其论域为 $C(p)$ 的经典结构 $\mathcal{C}(p)$。令 $\mathcal{L}(q)$ 为 $\mathcal{L}$ 的扩张，对于每个 $c \in C(q)$，该扩张是由在 $\mathcal{L}$ 中添加一个新常元得到。在模态逻辑中(第四章定义 2.2)，二元关系 S 可以是任意的，而在直觉主义逻辑中它总是设定为某个偏序 $\leqslant$。在这两种情形中，我们都设定各结构的论域具有单调性：如果 $pSq(p \leqslant q)$，那么 $C(p) \subseteq C(q)$。在直觉主

义逻辑中，我们还要求真原子事实集是单调的：如果 $p\leqslant q$，φ 是一个原子语句且 $\mathcal{C}(p)\vdash\varphi$，那么 $\mathcal{C}(q)\vdash\varphi$。

这里关键的定义（第四章定义 2.2 和第五章定义 2.2）是可能世界 p 和语句 φ 之间的*力迫关系*：$p\Vdash\varphi$。在两种逻辑对应的定义中，原子公式与对应于 $\vee$，$\wedge$ 和 $\exists$ 的归纳情形都是经典的真值定义（第二章定义 4.3）相同的"自然"推广：

如果 φ 是原子的，$p\Vdash\varphi\Leftrightarrow\mathcal{C}(p)\vdash\varphi$；$p\Vdash\varphi\vee\psi\Leftrightarrow p\Vdash\varphi$ 或 $p\Vdash\psi$；$p\Vdash\varphi\wedge\psi\Leftrightarrow p\Vdash\varphi$ 且 $p\Vdash\psi$；$p\Vdash\exists x\varphi(x)\Leftrightarrow$ 在 $\mathcal{L}(p)$ 中存在 c，使得 $p\Vdash\varphi(c)$。当然，模态逻辑的定义中还有给出 $\Box$ 和 $\Diamond$ 语义的条款。

然而模态逻辑和直觉主义逻辑的关键差别在于对 $\neg$，$\rightarrow$ 和 $\forall$ 的处理。在模态逻辑中，其解释继续遵循"自然"的经典风格，例如，$p\Vdash\neg\varphi\Leftrightarrow p$ 不力迫 φ。这种情形在直觉主义逻辑中是十分不同的。这里 $p\Vdash\neg\varphi$ 意思是没有 $q\geqslant p$ 力迫 φ。类似地，我们在直觉主义逻辑中对 $\rightarrow$ 和 $\forall$ 的定义为：

$p\Vdash\varphi\rightarrow\psi\Leftrightarrow$ 力迫 φ 的每个 $q\geqslant p$ 也力迫 ψ；

$p\Vdash\forall x\varphi(x)\Leftrightarrow$ 对每个 $q\geqslant p$ 和每个 $c\in\mathcal{L}(q)$，$q\Vdash\varphi(c)$。

一旦定义了力迫关系，在模态和直觉主义逻辑中，对应于经典真值（第二章定义 4.3）和永真性（第二章定义 4.4）的概念是相同的：*φ 在框架 $\mathcal{C}$ 中被力迫*，当且仅当对 $\mathcal{C}$ 中每个世界 p，有 $p\Vdash\varphi$；*φ 是永真的*，如果它在每个框架中都被力迫。

值得指出的是，对于 $\neg$，$\rightarrow$ 和 $\forall$ 的"非常规的"直觉主义解释都有些模态味道。例如，$p\Vdash\neg\varphi$ 和 $p\Vdash\forall x\varphi(x)$ 在直觉主义逻辑中的解释，很像 $p\Vdash\Box\neg\varphi$ 和 $p\Vdash\Box\forall x\varphi(x)$ 分别在模态逻辑中的解释。这些想法形成了从直觉主义逻辑到模态逻辑 S4 的哥德尔保持永真性变换。（见 Gödel [1933，2.3] 和 Gödel [1986，2.3] 的第 1 卷第 269 ~ 299 页的一些介绍性注记。）

三、表

原子表是为了反映不同联结词和量词的语义而设计的。因此原子语句和经典联结词的模态表（第四章定义 3.1）本质上与第二章第六节的经典表是相同的。例如，由 $T(\varphi\wedge\psi)$ 开始的经典表将 $T\varphi$ 和 $T\psi$ 连接在一起；由 $Tp\Vdash\varphi\wedge\psi$ 开始的模态表将 $Tp\Vdash\varphi$ 和 $Tp\Vdash\psi$ 连接在一起。

在经典逻辑中，仅用一个新常元作为真存在语句的证据（或者假全称语句的反例），在模态逻辑中除此之外，对于量词的分析还必须反映各论域之间的关系。因此举例来说，给定某表中某条路径上的一个表值 $Tp\Vdash\exists x\varphi(x)$，我们可以引入一个新 c，并在该路径上添加断言 $Tp\Vdash\varphi(c)$。另一方面，给定一个表值 $Tp\Vdash\forall x\varphi(x)$，我们可以这样展开它，对于任何适当的 c，添加 $Tp\Vdash\varphi(c)$，这里所谓"适当的 c"是指出现在任何关于某个世界 q 的断言中的任何 c，使得 p 是从 q 可达的（从而指定的论域 $C(p)$ 中任何世界都是从 q 可达的）。模态算子引入了建立新世界的可能性。例如，给定 $Tp\Vdash\Diamond\varphi$，我们引入一个新世界 q，使得 pSq 且 $q\Vdash\varphi$。我们还必须按照其语义和可达性关系解释 $\Box$：给定 $Tp\Vdash\Box\varphi$，对于任何适当的 q，即任何 q，使得 pSq，我们可以添加上 $Tq\Vdash\varphi$。

然而在直觉主义逻辑中，根据原子语句和联结词 $\neg$ 和 $\rightarrow$ 以及量词的新语义，我们必须做相应的改变。原子事实的单调性假设告诉我们，对于任何原子语句 φ，当给定 $Tp\Vdash\varphi$ 和 $p\leqslant q$ 时，添加上 $Tq\Vdash\varphi$。正如在 $\Box$ 的模态分析中所要求的那样，"非常规"联结词的分析涉及建立新世界或展望已定义的未来世界。例如，给定 $Tp\Vdash\neg\varphi$，对任何适当的 q，即任何使得 $p\leqslant$

q 的 q，我们可以添加 $Fq \Vdash \varphi$。另一方面，给定 $Fp \Vdash \neg\varphi$，我们可以引入一个新世界 q，使得 $p \leqslant q$，并添加断言 $Tq \Vdash \varphi$。对于直觉主义蕴涵的分析是类似的：如果 $Tp \Vdash \varphi \to \psi$，$p \leqslant q$ 且 $Tq \Vdash \varphi$ 都出现在某路径 P 上，那么我们可以添加 $Tq \Vdash \psi$ 到 P 上；如果 $Fp \Vdash \varphi \to \psi$ 出现，那么我们可以引入一个新的 q，并添加 $p \leqslant q$，$Tq \Vdash \varphi$ 和 $Fq \Vdash \psi$。量词表的构造，正如你可能期望的那样：给定 $Tp \Vdash \exists x\varphi(x)$，引入一个新的 c 并添加断言 $Tp \Vdash \varphi(c)$；给定 $Tp \Vdash \forall x\varphi(x)$，对于任何适当的 q 和 c，即任何 $q \geqslant p$ 和在世界 q 中的任何 c，我们可以添加 $Tq \Vdash \varphi(c)$。

一旦这些不同的语义已经构筑到原子表中，在模态逻辑（第四章定义 3.2）和直觉主义逻辑（第五章定义 3.2）中的*表定义*本质上与经典逻辑（第二章定义 6.1）中的定义是相同的。仅有的改变在于“新的”和“适当的”的概念，以使得它们符合相应的语义和术语。我们应当指出，如同在模态逻辑中那样，很可能在直觉主义逻辑中直接将序关系 $p \leqslant q$ 放在表中。我们没有这样做仅仅是因为，直觉主义逻辑中对于序关系构造的规定使得我们容易从表的其他部分读出该序关系。因此在仅用到直觉主义逻辑的工作中，采用经典样式而不在表中塞入序事实会比较简单。对于模态表中所构造的二元关系来说，其一般性使得直接记录这些关系更简单。

在任何情形中，*表证明*的概念（第二章定义 6.2，第四章定义 3.3，第五章定义 3.3）在三种逻辑中本质上是相同的：一个由断言 φ 不成立（或 φ 被力迫）开始的表是 φ 的一个证明，如果其中每条路径都含有矛盾（既断言某语句为真又断言它为假）。

四、可靠性和完全性

可靠性（第二章定理 7.2，第四章定理 4.3，第五章定理 4.3）和完全性（第二章定理 7.7，第四章定理 4.13，第五章定理 4.10）定理在三种逻辑中形式上是相同的：如果 φ 是可证的，那么它是永真的；如果 φ 是永真的，那么它是可证的。此外，至少在梗概上，可靠性和完全性的证明也是相同的。在可靠性的证明中，关键引理是：如果一个结构或框架与某表的根一致，那么在该表中存在一条路径，使得该结构或框架与该路径上每一个表值一致（第二章引理 7.1，第四章定理 4.2，第五章定理 4.2）。当然，经典逻辑中所使用的直白的“与……一致”概念（$\mathcal{A}$ 与 $T\varphi$ 一致，当且仅当 $\mathcal{A} \vdash \varphi$）必须进行推广，以处理框架和力迫，但是在模态（第四章定义 4.1）和直觉主义逻辑（第五章定义 4.1）中，这个概念在形式上是相同的。在这个基本引理中，对每一种情形的证明都是对表的构造进行归纳。模态和直觉主义论证的不同之处仍然仅仅是在不同的原子表上。每一个都遵循其相应的语义。给定这个基本引理，就直接得到可靠性的证明，且该证明在三种逻辑背景中是完全相同的：如果有 φ 的一个证明，那么该证明是从断言 φ 不成立开始的，并且表的每条路径包含矛盾。因为没有结构或框架可以与矛盾一致，所以没有结构或框架与该表的根一致，即 φ 是永真的。

完全性定理的证明是通过定义一个系统的方法，生成一个具有给定根的表，该表展开了每一个表值：*完全系统（模态或直觉主义）表*（CST，第二章定义 6.9；CSMT，第四章定义 4.8；CSIT，第五章定义 4.6）。在模态 CSMT 的构造中，还定义了一些可能世界和这些可能世界之间的可达性关系，这些定义都体现在其原子表中。在直觉主义情形中，可达性关系限制为偏序，这允许我们事先确定可能世界集，并为这个集合选取任何足够复杂的序。我们选取自然数的有穷序列集，其序为序列之间的扩展关系。在经典和模态表的展开中，每个表值都被展开的思想可以用*约化表值*和*完成表*（第二章定义 6.7，第四章定义 4.6）等概念来表

达。对于直觉主义逻辑，我们的证明没有给出明确的定义，而是把相应的要求放在其 CSIT 的构造和第五章引理 4.8 的证明中。(第五章定义 4.6 中，适当展开的表值概念取代了第四章定义 4.6 中的约化表值概念，且完成表概念被第五章引理 4.8 中所列性质所取代。)像第四章定义 4.6 中模态逻辑那样，给出直觉主义逻辑的相应定义也是合理的，尽管有些长。如果像上面所建议的那样，修改原子直觉主义表，把序关系直接放在表中，那么就可以像 CSMT 那样给出 CSIT 的定义，其中相应地调整了关于量词和“非常规”联结词的部分。

现在完全性定理在所有三种逻辑中的证明可以考虑使用从断言 φ 不成立或 φ 不被 p 所力迫开始的完全系统表。如果这个表不是 φ 的证明，那么它有一条非矛盾路径 P。现在用 P 上的表值建立一个与 P 上每个表值都一致(第二章定理 7.3，第四章定理 4.11，第五章定理 4.9)的结构或框架$\mathcal{C}$。在经典情形中，φ 在$\mathcal{C}$中不成立，而在模态和直觉主义情形中，p 在$\mathcal{C}$中不力迫 φ。因此，在每一种情形中 φ 都不是永真的，得证。

对于经典逻辑，我们定义单一的结构$\mathcal{C}$。其论域 C 由出现在 P 的断言中的所有项构成。对每个原子语句 φ，该结构本身定义为$\mathcal{C}\vDash\varphi$，当且仅当 $T\varphi$ 出现在 P 上。在模态和直觉主义两者的构造中，我们必须建立一个完整的框架$\mathcal{C}$。$\mathcal{C}$的可能世界集是由出现在 P 上断言中的 p 所构成的。在模态证明中，pSq 在$\mathcal{C}$中为真，当且仅当它出现在 P 上。直觉主义情形中的可达性关系事先由 P 上序列 p，q 的扩展关系所确定。然而，假如修改直觉主义原子表，让它直接记录序事实，我们就根据出现在 P 上的序事实来定义同样的序。

定义框架$\mathcal{C}$需要定义结构$\mathcal{C}(p)$的论域 $C(p)$。在两种情形中所定义的论域 $C(p)$是相同的，因为我们对这些论域作了同样的单调性假设：$C(p)$由$\mathcal{L}$中常元以及 P 上任何断言中出现的常元所构成，其中该断言含有某个可达 p 的 $q(q\leqslant p)$。在这些论域 $C(p)$上定义的结构$\mathcal{C}(p)$是不同的，其原因在于对直觉主义逻辑中原子事实的单调性假设。在模态情形中，仿照经典逻辑，我们定义一个原子语句 φ 在$\mathcal{C}(p)$中为真，当且仅当 $Tp\Vdash\varphi$ 出现在 P 上。在直觉主义构造中，$\mathcal{C}(p)\vDash\varphi$，当且仅当对任何 $q\leqslant p$，$Tq\Vdash\varphi$ 出现在 P 上。

对于这三种逻辑来说，论证我们刚才所定义的框架与 P 上每个表值一致，可以直接对出现在 P 上的语句 φ 的复杂性使用归纳法。这样可以得到完全性定理的证明。

五、逻辑后承

在经典和模态逻辑中，我们定义了某语句 φ 是某语句集 Σ 的逻辑后承的概念。经典地，$\Sigma\vDash\varphi$，当且仅当 φ 在任何使得 Σ 中每个语句 ψ 为真的结构$\mathcal{C}$中为真(第二章定义 4.4)。在模态逻辑中，我们说 φ 是 Σ 的逻辑后承，当且仅当 φ 在力迫每个 $\psi\in\Sigma$ 的框架$\mathcal{C}$中被力迫(第四章定义 2.8)。对于每一种逻辑，我们接着介绍从 Σ 出发的表证明概念。相对于标准表证明而言，唯一的改动在于，对于任何 $\psi\in\Sigma$ 与表上的任何路径 P(以及出现在 P 上的任何 p)，现在可以把 $T\psi(Tp\Vdash\psi)$添加到 P 上。(经典和模态情形分别见第二章定义 6.1 和第四章定义 3.11。)然后我们定义了从前件出发的完全系统表(第二章定义 6.9，第四章定义 4.14)，并证明相应的从前件出发的演绎的可靠性和完全性定理(第二章定理 7.2，第四章定理 4.13；第二章定理 7.7，第四章定理 4.15)。

尽管我们没明确地对直觉主义逻辑做相应的推广，但它们可以仿照模态逻辑的概念进行定义。我们把这个推广留作习题。

六、可判定性和希尔伯特式系统

在第五章第五节中，证明了命题直觉主义逻辑的可判定性(第五章定理 5.10)和全部谓

词直觉主义逻辑的不可判定性。在 Fitting [1983, 4.4]的第八章第七节中，有一个关于命题模态逻辑可判定性的基于表的证明。因为模态谓词逻辑包含经典逻辑，即不含模态算子的语句 φ 是经典永真的，当且仅当它是模态永真的(第四章第一节习题 10)，所以根据谓词逻辑的丘奇定理(第三章推论 8.10)，模态逻辑当然更是不可判定的。

在第四章第六节中，对于模态逻辑给出了一个希尔伯特式公理和规则系统。对于直觉主义逻辑描述一个对应的系统需要谨慎。因为$\neg$和$\rightarrow$不再构成联结词完全集，所以我们必须明确地处理每一个命题联结词。类似地，我们必须明确地给出关于$\exists$的公理和规则。Kleene [1952, 2.3]的第十九节给直觉主义逻辑提供了这样的系统，该系统有一个优点是，只要加上公理模式$\neg\neg\varphi\rightarrow\varphi$ 或 $\varphi\vee\neg\varphi$ 就可以扩展为经典逻辑的证明系统。

习题

如上面所建议的那样，我们严格地仿照我们对于模态逻辑所做的处理，给出直觉主义逻辑处理的一个梗概。

1. 重新表述直觉主义原子表，使得序关系可以明确地出现。例如，原子表(F→)变为：

$$\begin{array}{c} Fp\Vdash\varphi\rightarrow\psi \\ | \\ p\leqslant p' \\ | \\ Tp'\Vdash\varphi \\ | \\ Fp'\Vdash\psi \end{array}$$

2. 对于这个直觉主义表系统，描述对应的“新的”和“适当的”等概念，并证明其可靠性定理。
3. 对于这个系统，定义相应的约化表值和完成表等概念。
4. 定义对应的 CSIT 概念，并证明其完全性定理。
5. 像模态逻辑中所做的那样，定义直觉主义逻辑后承的概念。
6. 为了给出一般的从前件集出发的直觉主义表和对应的从前件出发的 CSIT 等概念，描述习题 2 ~ 4 中所需要的改变。
7. 描述习题 2 ~ 4 中所需要的改变，以证明直觉主义逻辑中从前件出发演绎的可靠性和完全性定理。

进一步阅读建议

给直觉主义逻辑描述一个如同第一章第七节和第二章第八节那样的希尔伯特式公理和规则系统需要谨慎。因为$\neg$和$\rightarrow$不再构成联结词完全集，我们必须明确地处理每一个命题联结词。类似地，我们必须明确地给出关于$\exists$的公理和规则。Kleene[1952, 2.3]的第十九节给直觉主义逻辑提供了这样的系统，该系统有一个优点是，只要加上公理模式$\neg\neg\varphi\rightarrow\varphi$ 就可以扩展为经典逻辑的证明系统。然后克林给出一个关于逻辑和递归论的详细论述，说明哪些定理已经被直觉主义地证明了，哪些还没有。

除了经典逻辑到直觉主义逻辑的永真性保持变换之外，Gödel [1933, 2.3]还提供了一个从直觉主义命题逻辑到(如第四章第六节中所描述的)模态逻辑 S4 的命题部分的保持永真性变换。该变换的解释与命题和谓词两个版本的有关证明的参考文献见 Gödel [1986, 2.3]第 1 卷第 296 ~ 299 页对于 Gödel [1933, 2.3]的介绍性评注。

Fitting [1983, 4.1]提供了许多不同的表方法。对于使用自然演绎和代数方法的直觉主

义逻辑所作的基本说明，见 van Dalen [1983，3.2]。对于直觉主义所作的哲学介绍，见 Dummett [1977，4.2]。关于构造性数学，见 Bridges 与 Richman [1985，4.2]、Bridges 与 Richman [1987，4.2]和 Troelstra 与 van Dalen [1988，4.2]。关于更高级的构造性数学的原数学课题，见 Beeson [1985，4.2]。

对于直觉主义类型论，见 Martin-Löf [1984，4.2]；它在一个计算机系统即构造性定理证明器上的应用，见 Constable[1986，5.6]。

最后，直觉主义"经典的"基础教材是 Heyting [1971，4.2]。

第六章　集合论基础

和其他现代数学教材一样，本书写作时假定读者已经具有了一些初等集合论的知识。但是这种假定常常是不合理的，因而，本章中将给出集合论的一种非形式化的叙述。在第一节到第六节，给出了集合论中的公理，并论述了初等集合论，这足以形式化数论包含的基本内容，例如，自然数作为一个满足著名的皮亚诺(Peano)后继公理的结构，在同构意义下是唯一的，而且通过归纳或者递归定义的函数也是存在并且唯一的。另外，还给出了本书其他章节所需要的一些基本的集合论概念，例如函数、序列和序。在第七节，阐述了一个基本的概念：有穷集合和可数集合的基数。余下的几节(第八节到第十一节)中，建立了超穷归纳原则，并提出了无穷序数和基数的基本理论、构成集合论的自然模型的累积分层以及选择公理的一些常见形式。同时，这些内容为计算机科学系或是数学系的任何研究生课程提供了足够的集合论背景。本章的内容独立于本书其他章节，与第一章第一节和附录 A 的一小部分内容有所重复。

第一节　集合论中的一些基本公理

集合论最基本的概念是相等 = 和隶属关系 ∈ (通常把它们的否定分别记为 ≠ 和 ∉)。集合也可以用作变量。特别地，集合的元素仍然是集合。我们首先用非形式化的语言列出集合论中最基本的公理。

公理 1(外延性)　*假设 y，z 都是集合，且对于所有的 x，$x \in y$，当且仅当 $x \in z$。则 $y = z$。*

因此，一个集合完全是由它的元素确定的。

公理 2(空集)　*存在一个集合 x，满足对所有的 y，$y \notin x$。*

因此，存在一个不包含任何元素的集合。根据公理 1，这样的集合是唯一存在的，我们称之为空集(empty set)，并记作 $\varnothing$。

公理 3(无序对)　*假设 x 和 y 都是集合。那么，存在一个集合 z，满足对所有的 w，$w \in z$，当且仅当 $w = x$ 或者 $w = y$。*

因此，对任何集合 x 和 y，存在一个集合 z，它仅有两个元素：x 和 y。根据公理 1，z 由 x 和 y 唯一确定。这个集合称为 x 和 y 的*无序对*(unordered pair)，记作 $\{x, y\}$。

公理 4(并集)　*假设 x 是一个集合，那么存在一个集合 y，满足对所有的 z，$z \in y$，当且仅当存在一个 u，满足 $z \in u$ 且 $u \in x$。*

也就是说，对任何集合 x，存在一个集合 y，它的元素正好是所有 x 的元素的元素。根据公理 1，这个集合 y 是唯一确定的。我们称之为 x 的*并集*，并记作 $\cup x$。

公理 5(子集构造，也称为分离或概括)　*假设 $P(x)$ 是集合 x 的一个性质，且 z 是一个集合。则存在一个集合 y，满足对任何的 x，$x \in y$，当且仅当 $x \in z$ 和 $P(x)$ 成立。*

根据公理 1，y 由 P 和 z 唯一确定，记作 $\{x \in z \mid P(x)\}$。

评注 1.1

(i)公理 2 指出至少存在一个集合，即空集。

(ii)公理3、公理4和公理5可以看作是集合的“生成”规则。它们使我们能从已有的集合构造新的集合。

(iii)根据公理2、公理3、公理4和公理5生成的集合都是有穷集合。(参见习题5。)

(iv)公理5最早是由Zermelo[1908, 2.3]提出的，文中列出了集合论中一些描述“明确的”性质的公理。Fraenkel[1922, 2.3]和Skolem[1922, 2.3]等提出“明确的”性质这个概念并不够准确，因此，他们认为公理5是含糊不清的。另一种观点认为使用“性质”最广泛的解释，就不会产生任何矛盾，何必杞人忧天？特别地，子集构造本身并不会像构造所有集合的集合那样导致著名的罗素悖论——Russell[1903, 2.3]。

罗素悖论

如果所有集合也构成一个集合，记作w，对w和性质$P(x)$(定义为$x\notin x$)应用公理5，可以得到一个新的集合y。当我们考虑y是否是它自身的元素时，悖论就产生了。如果$y\in y$，则由于y是w的子集，故$P(y)$成立，从而有$y\notin y$。另一方面，如果$y\notin y$，则根据P的定义，$P(y)$成立，由于y是w的子集，有$y\in y$。无论哪种情形，都可以导出矛盾。我们稍后将看到，和其他的悖论(例如，定理8.6)一样，在朴素集合论中，解决这种明显的矛盾的方法是认为包含所有集合的集合w是不存在的。特别地，我们肯定不能从公理5推导出w的存在性，因为子集构造公理中，新构造的集合元素只允许从前面已经给出的集合中选择。

为了准确描述“明确的”性质的概念，Fraenkel[1922, 2.3]主张将可以接受的集合性质限制到可以用集合上特定的函数迭代得到的性质。Skolem[1922, 2.3]主张只允许由$=$和$\in$构造的带有集合变量的谓词逻辑公式所定义的性质。后者被现代公理集合论广泛采用，它本身是谓词逻辑中的一个理论(见第六节)。

评注1.2 我们常常使用标准的符号来描述一个集合x，说“x存在”是指存在一个集合满足关于x的描述(根据集合论公理，可以加以证明)。例如，可以说集合$\{\varnothing\}$存在，表示存在一个只含有元素$\varnothing$的集合。同样地，集合$\{y \mid y$的每一个元素都是x的元素$\}$存在，表示存在一个集合，它的元素正好是那些满足“每一个y的元素也是x的元素”的y。当然，如果给出一个可以确定x的元素的描述，则由外延性公理，满足给定描述的集合至多有一个。因而，集合的存在性证明同样给出了它的唯一性。

习题

1. 证明：对每个x，y，$\{x, y\}=\{y, x\}$。
2. 证明下面的集合都是存在的：
   (a)$\{\varnothing\}$
   (b)$\{\varnothing, \{\varnothing\}\}$
   (c)$\{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}\}$
   (d)$\{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\varnothing, \{\varnothing\}\}\}$。
3. 证明：对每个x，y，z，下列集合都是存在的：
   (a)$\{x\}$
   (b)$\{x, y, z\}$
   (c)$x\cup y$，集合w满足$a\in w$，当且仅当$a\in x$或$a\in y$。
4. 证明：如果x是一个有穷集合，那么x的幂集(x的所有子集组成的集合)$\mathcal{P}(x)=\{z \mid z$的每一个元素都是$x$的元素$\}$也是一个集合，而且是有穷集合。(提示：对$x$的元素的个数进行归纳证明。)
5. 在自然数集$\mathcal{N}$上归纳定义集合$R(n)$如下：$R(0)=\varnothing$，$R(n+1)=\mathcal{P}(R(n))$，令$R=\bigcup\{R(n) \mid n\in\mathcal{N}\}$。

证明下列命题：

(a)每个 $R(n)$ 都是有穷的。

(b)在 R 内，公理 2 到公理 5 都成立，即 $\varnothing \in R$ 且对每个 $x, y \in R$，$\{x, y\}$，$\cup x \in R$。

(c) R 包含在任何满足(b)的集合之中。

(这些习题的证明都是简单的数学问题，集合论中这些思想的形式化内容可在第四节习题 1 和第六节习题 1 中找到，更一般地详见第十节。)

第二节 集合的布尔代数

在给出集合论的更多公理之前，我们使用公理 1 ~ 5 来非形式化地描述集合的布尔代数 [1847, 1854, 2.3]。在布尔代数中，有三种基本的集合运算：并(union)、交(intersection)和差(difference)。还有一个重要的概念是包含关系。

定义 2.1(并) *两个集合 A 和 B 的并是集合 $A \cup B$，它的元素正好是 A 或者 B 的元素。*

公理 3 和公理 4 保证了并集的存在性。任意给定两个集合 A 和 B，集对 $\{A, B\}$ 是由公理 3 得到的，且由等式两边的定义有 $A \cup B = \cup\{A, B\}$。公理 1 保证了这个集合的唯一性，我们称之为 A 和 B 的并。用归纳法可以简单地证明：能够构造任何有穷多个集合的并(留作习题 1)。这段话同样适用于下面定义的集合的交。

我们使用子集构造公理来构造两个集合 A 和 B 的交和差(或称为相对补(relative complement))，分别记作 $A \cap B$ 和 $A - B$。

定义 2.2(交) *令 $P(x)$ 表示性质 $x \in B$。则 $A \cap B = \{x \mid x \in A$ 且 $P(x)\} = \{x \mid x \in A$ 且 $x \in B\}$。*

定义 2.3(差) *令 $P(x)$ 表示性质 $x \notin B$。则 $A - B = \{x \mid x \in A$ 且 $P(x)\} = \{x \mid x \in A$ 且 $x \notin B\}$。*

定义 2.4(包含) *我们称 A 是 B 的**子集**(subset)，记为 $A \subseteq B$，如果对于所有的 x，$x \in A$ 蕴涵 $x \in B$。*

外延性公理说明：两个集合是相等的，当且仅当它们相互包含。也就是说，要证明两个集合相等，一种方法就是证明它们的相互包含关系。在布尔的集合代数中，他选定了一个集合 U 作为"全集"。他从上面的并、交和差的定义出发，推导出了这些运算的代数性质。下面列出 U 的子集 A，B，C 上的一些基本运算定律。我们用 $\neg A$ 表示 A 的补(complement)：$U - A$。

结合律(Associativity Law)

(1A) $(A \cup B) \cup C = A \cup (B \cup C)$　(1B) $(A \cap B) \cap C = A \cap (B \cap C)$

交换律(Commutativity Law)

(2A) $A \cup B = B \cup A$　(2B) $A \cap B = B \cap A$

幂等律(Idempotence Law)

(3A) $A \cup A = A$　(3B) $A \cap A = A$

分配律(Distributivity Law)

(4A) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

(4B) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

德·摩根律(De Morgan's Law)

(5A) $\neg(A \cap B) = (\neg A) \cup (\neg B)$　(5B) $\neg(A \cup B) = (\neg A) \cap (\neg B)$

否定律(Negation Law)

(6A)$A\cup(\neg A)=U$　(6B)$A\cap(\neg A)=\varnothing$

零律(Empty Set Law)

(7A)$A\cup\varnothing=A$　(7B)$A\cap\varnothing=\varnothing$

吸收律(Absorption Law)

(8A)$A\cup U=U$　(8B)$A\cap U=A$

双重否定律(Double Negation Law)

(9)$A=\neg(\neg(A))$

我们简述三种证明这些运算定律的方式。第一种方式是通过证明两个集合互为子集来说明两者相等。第二种方式是直接应用外延性公理，这就是在同一个论证中证明所给假设的“当且仅当”条件，而这种方法每一步使用的都是给定的定义或者通常的演绎规则。如果已经学过了谓词逻辑，则可以使用第三种方式：应用相应的逻辑定律证明。最后一种方式清晰地指出了罗素的观点：数学问题(至少集合论)可以归约到逻辑问题。下面以分配律(4B)为例，证明如下。

命题 2.5　$A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$。

我们用三种方式来证明。

证明　方式1。首先证明(Ⅰ)$A\cap(B\cup C)\subseteq(A\cap B)\cup(A\cap C)$；然后证明(Ⅱ)$(A\cap B)\cup(A\cap C)\subseteq A\cap(B\cup C)$。再根据外延性公理得到(4B)。

Ⅰ. 假设$x\in A\cap(B\cup C)$。根据交集的定义，$x\in A$且$x\in B\cup C$。根据并集的定义，后者蕴涵了$x\in B$或者$x\in C$。

情形1：$x\in B$。此时，由于$x\in A$且$x\in B$，故$x\in A\cap B$(交集的定义)，从而$x\in(A\cap B)\cup(A\cap C)$(并集的定义)。

情形2：$x\in C$。根据假设$x\in A$，则$x\in A\cap C$(交集的定义)，$x\in(A\cap B)\cup(A\cap C)$(并集的定义)。

Ⅱ. 假设$x\in(A\cap B)\cup(A\cap C)$。根据并集的定义，或者$x\in A\cap B$，或者$x\in A\cap C$。

情形1：$x\in A\cap B$。此时，$x\in A$且$x\in B$(交集的定义)，后者蕴涵了$x\in B\cup C$(并集的定义)。从而，$x\in A\cap(B\cup C)$(交集的定义)。

情形2：$x\in A\cap C$。此时，$x\in A$且$x\in C$(交集的定义)，后者蕴涵了$x\in B\cup C$(并集的定义)。从而，$x\in A\cap(B\cup C)$(交集的定义)。

命题得证。

方式2。$x\in(A\cap(B\cup C))$，当且仅当$(x\in A)$且$(x\in(B\cup C))$，当且仅当$(x\in A)$且$((x\in B)$或$(x\in C))$，当且仅当$((x\in A)$且$(x\in B))$或$((x\in A)$且$(x\in C))$，当且仅当$(x\in(A\cap B))$或$(x\in(A\cap C))$，当且仅当$x\in((A\cap B)\cup(A\cap C))$。

方式3。了解谓词逻辑的读者可以看出，这种方式可以看作是方式2的形式化证明。每个恒等式简单地归约为对应的谓词逻辑的永真语句和外延性公理的一个应用。对于分配律，根据外延性公理，只要证明

$$(\forall x)[x\in A\cap(B\cup C)\leftrightarrow x\in((A\cap B)\cup(A\cap C))] \qquad (*)$$

如果引入谓词逻辑语言，它带有三个一元谓词符号$A(x)$、$B(x)$和$C(x)$，分别表示$x\in A$、

$x \in B$ 和 $x \in C$，则可以用表证明下面的语句：

$$(\forall x)[(A(x) \wedge (B(x) \vee C(x))) \leftrightarrow ((A(x) \wedge B(x)) \vee (A(x) \wedge C(x)))] \quad (*')$$

上面需要证明的($*$)，仅仅是这个永真语句的一个实例。 □

所有的恒等式都可以用上面的任意一种方式来验证(留作习题2)。

习题

1. 对 n 用归纳法证明：如果 x_1，…，x_n 都是集合，则 $x_1 \cup \cdots \cup x_n$ 和 $x_1 \cap \cdots \cap x_n$ 也都是集合。

2. 验证集合上的每一个布尔运算定律：(1A)～(9)。

第三节　关系、函数和幂集公理

本节由有序对的定义开始给出一些初等数学的基本定义。这不需要新的公理。

定义 3.1　*如果 x 和 y 都是集合，则***有序对**(ordered pair)$\langle x, y\rangle$定义为集合$\{\{x\}, \{x, y\}\}$。称 x 和 y 分别为这个有序对的**第一个坐标**和**第二个坐标**。定义**有序三元组**(ordered triple)$\langle x_1, x_2, x_3\rangle$为$\langle\langle x_1, x_2\rangle, x_3\rangle$，并且通过归纳，将**有序 n 元组**(ordered n-tuple)$\langle x_1, x_2, \cdots, x_n\rangle$定义为$\langle\langle x_1, x_2, \cdots, x_{n-1}\rangle, x_n\rangle$。这些 n 元组的不同坐标也通过归纳定义。

注意到$\{x\} = \{x, x\}$，所以两次应用无序对公理可以证明有序对$\langle x, y\rangle$是存在的。通过归纳和重复应用这条公理，可以证明有序 n 元组存在(习题1)。下面是有序对的基本性质。

命题 3.2　*如果 x, y, z, w 都是集合，则$\langle x, y\rangle = \langle z, w\rangle$，当且仅当 $x = z$ 且 $y = w$。类似地，$\langle x_1, x_2, \cdots, x_n\rangle = \langle y_1, y_2, \cdots, y_n\rangle$，当且仅当 $x_1 = y_1$，$x_2 = y_2$，…，$x_n = y_n$。*

证明　当然，如果 $x = y$ 且 $y = w$，那么$\langle x, y\rangle = \langle z, w\rangle$。反过来，如果$\langle x, y\rangle = \langle z, w\rangle$，那么$\{\{x\}, \{x, y\}\} = \{\{z\}, \{z, w\}\}$。如果 $x = y$，那么$\langle x, y\rangle = \{\{x\}, \{x, y\}\}$恰有一个元素，因为由外延性公理有$\{x, x\} = \{x\}$。在这种情况下，$\{\{z\}\{z, w\}\} = \{\{x\}\}$，所以$\{z\} = \{z, w\} = \{x\}$。再由外延性有 $x = z = w$。另一方面，如果 $x \neq y$，那么$\langle x, y\rangle$(因此$\langle z, w\rangle$)恰有两个元素：$\{x\}$和$\{x, y\}$($\{z\}$和$\{z, w\}$)。$\langle x, y\rangle$的每个元素的元素是唯一的集合 x，同时$\langle z, w\rangle$的每个元素的元素是唯一的集合 z。因此 $x = z$。最后，$\langle x, y\rangle$($\langle z, w\rangle$)的一个元素的元不同于 x 的唯一集合是 $y(w)$，所以 $y = w$。

对 n 元组的证明留作习题2。 □

为了定义两个集合的笛卡儿积，并说明它也是一个集合，我们需要集合论中的另一个公理。

公理 6(幂集)　*假设给定集合 x，则存在集合 y，它的元素恰是 x 的所有子集。*

由外延性公理，对每个 x 恰有一个这样的 y，常被记为$\mathcal{P}(x)$，称为 x 的**幂集**(power set)。

例 3.3

(i)如果 $A = \{x, y\}$，那么$\mathcal{P}(A) = \{\varnothing, \{x\}, \{y\}, \{x, y\}\}$。

(ii)对任何集合 A，有$\varnothing \in \mathcal{P}(A)$而且 $A \in \mathcal{P}(A)$。

(iii)如果 A 是一个含有 n 个元素的有穷集，那么$\mathcal{P}(A)$是一个有 2^n 个元素的有穷集。

假设给定集合 A 和 B。应用无序对公理和并集公理构造出 $A \cup B$。由幂集公理(公理6)，$\mathcal{P}(A \cup B)$存在。再由幂集公理，$\mathcal{P}(\mathcal{P}(A \cup B))$存在。现在对这个集合应用子集构造公理，并利用如下性质 Q：$Q(x)$成立，当且仅当 $x = \langle a, b\rangle$，对某个 $a \in A$ 和某个 $b \in B$。则

$$\{x \mid x \in \mathcal{P}(\mathcal{P}(A \cup B)) \text{ 且 } Q(x)\}$$

是一个集合。

这个集合称为 A 和 B 的笛卡儿积(Cartesian product)，记为 $A\times B$。因此，

$$A\times B=\{\langle a,b\rangle \mid a\in A \text{ 且 } b\in B\}$$

通过归纳，定义有限次笛卡儿积 $A_1\times A_2\times\cdots\times A_n$ 为 $(A_1\times A_2\times\cdots\times A_{n-1})\times A_n$，即对每个 $i\leqslant n$，满足 $x_i\in A_i$ 的所有 n 元组 $\langle x_1, \cdots, x_n\rangle$ 的集合。(见习题3。)记 A 和它自身 n 次的笛卡儿积为 A^n。

定义 3.4 一个**二元关系**(binary relation)是一个有序对的集合。对一个二元关系 R，常用 aRb 表示 $\langle a, b\rangle\in R$。这称为**关系符号**(relational notation)。R 的**第一(坐标)投影**(first (coordinate) projection)是 $\{a \mid$ 对某个 b，$aRb\}$。R 的**第二(坐标)投影**(second (coordinate) projection)是 $\{b \mid$ 对某个 a，$aRb\}$。n 元关系和相应的投影可通过替换有序对为 n 元组类似地定义。另一个表示 n 元关系 R 对 n 元组 $\langle a_1, \cdots, a_n\rangle$ 成立的常用记号是 $R(a_1, \cdots, a_n)$ (意思是 $\langle a_1, \cdots, a_n\rangle\in R$)。

Peirce[1870, 2.3]发明了关系代数，它是现代谓词逻辑的前身。由于关系是集合，故并、交和补的布尔运算适用于关系。他引进了另外两个运算：逆(converse) $\breve{R}$ 和复合(composition) $S\times R$。下面是用关系符号表示的五种二元关系上的运算。

$a(R\cup S)b$，当且仅当(aRb 或 aSb)。

$a(R\cap S)b$，当且仅当(aRb 且 aSb)。

$a(R-S)b$，当且仅当(aRb 且非 aSb)。

$a(\breve{R})b$，当且仅当 bRa。

$a(S\times R)b$，当且仅当存在一个 c，满足 aSc 且 cRb。

对这些运算有一组更加精细的代数定律。最重要的是复合运算的结合律：$T\times(S\times R)=(T\times S)\times R$。

命题 3.5 如果 T，S 和 R 是二元关系，那么 $T\times(S\times R)=(T\times S)\times R$。

证明 通过反复应用两个二元关系乘积($\times$)的定义，证明：对任何 a 和 b，$a(T\times(S\times R))b$ 成立，当且仅当 $a((T\times S)\times R)b$ 成立。首先，$a(T\times(S\times R))b$ 成立，当且仅当存在一个 c 满足 aTc 且 $c(S\times R)b$，当且仅当存在一个 c 满足 aTc 且存在一个 d 满足 cSd 且 dRb。类似地，$a((T\times S)\times R)b$，当且仅当存在一个 d 满足 $a(T\times S)d$ 且 dRb，当且仅当存在一个 d 且存在一个 c 满足 aTc 且 cSd 且 dRb。 □

我们把许多类似性质的验证留作习题 4～5。

定义 3.6 R 是一个**从 A 到 B 的二元关系**，如果 R 是 $A\times B$ 的一个子集。R 是一个 **A 上的二元关系**，如果 R 是 $A\times A$ 的一个子集。一个 A 上的 **n 元关系**是 A^n 的一个子集。

定义 3.7 一个集合 A 上的关系 R 称为：

(i)**自反的**(reflexive)，如果对所有 $x\in A$，xRx。

(ii)**对称的**(symmetric)，如果对所有 x，$y\in A$，xRy 蕴涵 yRx。

(iii)**传递的**，如果对所有 x，y，$z\in A$，xRy 且 yRz 蕴涵 xRz。

(iv)**反对称的**(antisymmetric)，如果对所有 x，$y\in A$，xRy 蕴涵非 yRx。

(v)**反自反的**，如果对所有 $x\in A$，非 xRx。

(vi)一个**等价关系**，如果 R 是自反的、对称的且传递的。

如果 R 是 A 上的一个等价关系，那么对每个 $x\in R$，集合 $\{y \mid yRx\}$ 称为 R 的等价类(equivalence class)。我们称 $\{y \mid yRx\}$ 为 x 的等价类，并记作 $[x]$。

例 3.8

(i)关系≤(小于或等于)在自然数集$\mathcal{N}=\{0, 1, 2, \cdots\}$上是自反的且传递的。关系<(严格小于)在$\mathcal{N}$上是反自反的、传递的且反对称的。≤的第一投影和第二投影与<的第一投影一样都是$\mathcal{N}$。<的第二投影是$\mathcal{N}-\{0\}$。

(ii)关系⊆(包含)在$\mathcal{P}(\mathcal{N})$上是自反的和传递的。关系⊂(真包含,定义为$A\subset B$,当且仅当$A\subseteq B$且$A\neq B$)在$\mathcal{P}(\mathcal{N})$上是反自反的、传递的和反对称的。⊆的第一投影和第二投影都是$\mathcal{P}(\mathcal{N})$。⊂的第一投影是$\mathcal{P}(\mathcal{N})-\{\mathcal{N}\}$,第二投影是$\mathcal{P}(\mathcal{N})-\{\varnothing\}$。

(iii)关系=在任何集合A上都是等价关系。

(iv)关系|(这里$a\mid b$是指a整除b且$a\neq b$)在集合$\mathcal{N}-\{1\}$上是反自反的、传递的和反对称的。这个关系的第一投影是$\mathcal{N}-\{0, 1\}$,第二投影是$\{x\in\mathcal{N}\mid x$不是素数$\}$。

(v)给定任何$n\in\mathcal{N}$,在整数$\mathcal{Z}=\{0, \pm1, \pm2, \cdots\}$上定义如下关系:$a\equiv b(\bmod n)$,当且仅当对某个$k\in\mathcal{Z}$,$b-a=kn$。这是一个$\mathcal{Z}$上的等价关系。

(vi)对于$\mathcal{N}$上的等价关系=,等价类是单点集$\{0\}$,$\{1\}$,…。对于$\mathcal{Z}$上的关系$\equiv(\bmod 3)$,等价类是

$[0]=\{0, 3, 6, 9, \cdots\}\cup\{-3, -6, -9, \cdots\}$

$[1]=\{1, 4, 7, \cdots\}\cup\{-2, -5, -8, \cdots\}$

$[2]=\{2, 5, 8, \cdots\}\cup\{-1, -4, -7, \cdots\}$

注意,在最后一个例子(例3.8(vi))中,$[3]=[0]$,$[4]=[1]$,等等。所以只存在三个不同的等价类。这个例子的各种结论的验证留作习题6。

定义 3.9 一个非空集合A的**划分**(partition)P是A的子集的集合,满足:

(i)$\varnothing\notin P$

(ii)$\bigcup P=A$

(iii)对任何P_1,$P_2\in P$,满足$P_1\neq P_2$,则有$P_1\cap P_2=\varnothing$。

命题 3.10

(i)设R是非空集合A上的一个等价关系。则R的所有等价类的集合是A的一个划分。

(ii)设P是A的一个划分。定义A上的一个二元关系R为xRy,当且仅当在P中存在一个P_1,满足$x\in P_1$且$y\in P_1$。则R是A上的一个等价关系。

证明 (i)因为R是自反的,故对任何$x\in A$,$x\in[x]$。因此,划分定义中的(i)和(ii)成立。现在假设$[x]\cap[y]\neq\varnothing$。则对某个$z\in A$,有$z\in[x]$且$z\in[y]$,因而$zRx$,$zRy$,这意味着(由对称性和传递性)$xRy$。但$w\in[x]$,当且仅当$wRx$,当且仅当$wRy$,当且仅当$w\in[y]$,所以$[x]=[y]$。

(ii)这部分的证明留作习题9。 □

定义 3.11 一个**偏序**是一个有序对$\langle A, R\rangle$,其中A是一个集合,且R是A上反自反的和传递的二元关系。如果$\langle A, R\rangle$是一个偏序,通常称R是A的一个**偏序**(或**偏序化**(partially orders)A),并且记xRy为$x<_R y$(当关系R在上下文中含义明确时,可以简单记作<)。如果<是一个偏序的二元关系,那么像平常那样用记号$x\leqslant y$、$x>y$和$x\geqslant y$分别表示($x<y$或$x=y$)、($y<x$)和($y<x$或$y=x$)。

定义 3.12 一个**线性序**是一个偏序$\langle A, R\rangle$,满足对任意的x,$y\in R$,或者xRy,或者

$x=y$，或者 yRx。一个序的这种性质，即三种可能性必有一种成立，称为**三分法**(trichotomy)。(在这种情况下，称 R **线性序化**(linearly orders)A 或是 A 的一个**线性序**。)

注意到，如果$\langle A, R\rangle$是一个偏序，那么 R 是反对称的，因此如果它是一个线性序，则由三分法给出的三个关系中必然有一个成立(习题7)。

例 3.13

(i)$\langle \mathcal{N}, <\rangle$是一个线性序，其中 $x<y$ 就是通常所说的 x 严格小于 y。

(ii)$\langle \mathcal{N}-\{0\}, \mid\rangle$是一个偏序，其中 $\mid$ 和上面例 3.8(iv)中的定义是一样的。

(iii)$\langle \mathcal{P}(\mathcal{N}), \subset\rangle$是一个偏序。

定义 3.14(函数) 一个二元关系 f 称为**函数**(function)，如果对每个 a，b_1 和 b_2，afb_1 且 afb_2 蕴涵了 $b_1=b_2$。当 afb，即$\langle a, b\rangle\in f$ 时，记 $f(a)=b$。这称为**函数记号**(functional notation)。函数 f 的第一投影，即$\{x \mid$ 存在一个 y，满足 $f(x)=y\}$，称为 f 的**定义域**，记为 dom f。f 的第二投影，即$\{y \mid$ 存在一个 x，满足 $f(x)=y\}$，称为 f 的**值域**(range)，记为 rg f。我们视 $n>1$ 的 n 元函数为有序 n 元组的集合上的函数。

定义 3.15(映射) 一个**映射**(map)是一个有序三元组$\langle f, A, B\rangle$，由函数 f、集合 A 和集合 B 组成，其中，f 的定义域是 A 且值域是 B 的一个子集。映射记为 $f: A\to B$，这个记号读作"f 把 A 映入 B"。A 称为映射$\langle f, A, B\rangle$的**定义域**，B 称为**陪域**(codomain)。

注意，一个函数可以产生很多不同的映射，因为映射的陪域可以是任何包含 f 的第二投影的集合。

定义 3.16

(i)一个函数 f 称为**一一的**(one-to-one)(也写作 1-1)，如果 $f(a_1)=f(a_2)$ 蕴涵着 $a_1=a_2$。一个映射 $f: A\to B$ 是 1-1 的，如果函数 f 是一一的。一个一一的函数(或映射)也称为**单射**(injection)。

(ii)一个映射 $f: A\to B$ 称为**映满的**(onto)，如果对每个 $b\in B$，存在一个 $a\in A$，满足 $f(a)=b$。这个记号不适用于函数，仅适用于映射。一个映满的映射也称为**满射**(surjection)。

(iii)一个映射 $f: A\to B$ 称为**一一对应**(one-to-one correspondence)，如果映射 f 既是 1-1 的又是映满的。这样的映射也称为**双射**(bijection)。

(iv)如果 $f: A\to B$，$f\subseteq g$ 且 $g: A'\to B$，则 f 称为 g 的一个**限制**(restriction)。g 称为 f 的**扩张**(extension)。一个映射 $g: A'\to B$ 在集合 $A\subseteq A'$上的限制为映射 $f: A\to B$，对 $x\in A$，满足 $f(x)=g(x)$。我们记这个限制为 $g\upharpoonright A$。

(v)如果 $f: A\to B$ 是一个单射，那么它的逆 f^{-1} 是从 rg f 到 A 的映射，由 $\breve{f}$ 在 rg f 上的限制给出。(见习题 8(e)。)

要证一个映射 $g: A'\to B$ 在集合 $A\subseteq A'$上的限制 f 存在，只需令 $f=g\cap(A\times B)$。由外延性，这个限制是唯一的。

定义 3.17 一个集合 A 上的**二元运算**(binary operation)是一个函数 $f\subseteq(A\times A)\times A$，满足 dom $f=A\times A$ 且 rg $f\subseteq A$。对于二元运算，我们通常记为 afb，而不是 $f(\langle a, b\rangle)$。这就是说，我们写成 $a+b$ 而不是 $+(a, b)$。这是**运算**(operational)或**中缀**(infix)记号，而不是函数或**前缀**(prefix)记号。

定义 3.18 如果 $f: A\to B$ 和 $g: B\to C$ 都是映射，那么 f 和 g 的**复合**(composition)$g\circ f$

是从 A 到 C 的映射，满足 $(g\circ f)(x)=y$，当且仅当对某个 z，$f(x)=z$ 且 $g(z)=y$。因此，$(g\circ f)(x)=g(f(x))$，并且我们经常记 $g\circ f$ 为 gf。

如果将 f 和 g 看成关系，那么它们作为映射和作为关系的复合是一致的(见习题 8(a))。

在本节的最后，给出集合 B^A 的构造，它是所有以 A 作为定义域且值域包含在 B 中的函数的集合。假设给定集合 A 和 B。运用笛卡儿积、幂集公理和子集构造公理，可以构成集合

$$\{z\in\mathcal{P}(A\times B)\mid z\text{ 是一个定义域为 }A\text{ 的函数}\}。$$

这就是集合 B^A。

习题

1. 证明(通过对 n 归纳)：对任意集合 $x_1,\cdots,x_n$，有序 n 元组 $\langle x_1,x_2,\cdots,x_n\rangle$ 存在。

2. 证明：对每个 n，以及任何集合 $x_1,x_2,\cdots,x_n$ 和 $y_1,y_2,\cdots,y_n$，有序 n 元组 $\langle x_1,x_2,\cdots,x_n\rangle$ 和 $\langle y_1,y_2,\cdots,y_n\rangle$ 相等，当且仅当 $x_1=y_1$，$x_2=y_2$，$\cdots$，$x_n=y_n$。

3. 证明：对每个 n，$A_1\times\cdots\times A_n$ 是所有 n 元组 $\langle x_1,x_2,\cdots,x_n\rangle$ 的集合，对每个 i 满足 $1\leqslant i\leqslant n$，有 $x_i\in A_i$。

4. 假设 R、S 和 T 是集合 X 上的任何二元关系，$I=\{\langle x,x\rangle\mid x\in X\}$ 是 X 上的恒等关系并且 $U=\{\langle x,y\rangle\mid x,y\in X\}$ 是 X 上的泛关系。

(a)证明：上一节中每一个集合的布尔运算定律对关系 R、S 和 T 成立。(再次假设 $\neg R$ 表示 $U-R$。)

(b)证明：$R\times I=R=I\times R$。

(c)证明：$\breve{U}=U$ 且 $\breve{I}=I$。

5. 证明下列关于二元关系 R、S 和 T 的定律或找出反例：

(a)$\breve{R}$ 的逆是 R。

(b)$R\cap(S\times T)=(R\cap S)\times(R\cap T)$。

(c)$R\cup(S\times T)=(R\cup S)\times(R\cup T)$。

6. 证明：例 3.8 中关于关系的各种断言(i)~(vi)。

7. 证明：如果 $\langle A,<\rangle$ 是一个偏序，那么 $<$ 是反对称的。证明：如果 $\langle A,<\rangle$ 是一个线性序且 $x,y\in A$，那么关系 $x<y$，$x=y$，$y<x$ 之一恰好成立。

8. 假设 f 和 g 是映射(因而也是二元关系)。

(a)证明：$f\circ g=f\times g$(其中 $\times$ 是上一节中定义在二元关系上的运算)。

(b)证明：如果 f 是 1-1 的，那么 $\breve{f}$ 是一个函数。它的定义域和值域是什么？

(c)证明：如果 f 是一个双射，那么 $\breve{f}$ 也是一个双射。

(d)证明：如果 f 和 g 都是双射，那么 $f\circ g$ 也是双射。

(e)证明：如果 f 是一个双射，那么 $f^{-1}=\breve{f}\restriction \mathrm{rg}\,f$($f$ 的逆限制在 f 的值域上)是一个函数，对每个 $x\in\mathrm{dom}\,f$，满足 $f^{-1}\circ f(x)=x$，且对每个 $y\in\mathrm{rg}\,f$，满足 $f\circ f^{-1}(y)=y$。如果 f 是一个双射，那么 $f^{-1}=\breve{f}$。

9. 证明命题 3.10(ii)。

10. 证明：对任何 A,B,C，映射 K：$B^A\times C^A\to(B\times C)^A$ 是一个双射，满足对每个 $a\in A$，$K(f,g)(a)=\langle f(a),g(a)\rangle$。

11. 证明：$A\times B=\varnothing$，当且仅当 $A=\varnothing$ 或 $B=\varnothing$。

第四节　自然数、算术和无穷

根据公理 3~6，可以由公理 2 给出的空集构造出新的集合。通过有穷次应用这些公理所构造出的集合都将是有穷的(习题 1)。直观上看，我们注意到两个有穷集合的无序对是有穷的，有穷集合的有穷次并是有穷的，任何有穷集合的子集是有穷的，并且有穷集合的幂集

也是有穷的。于是由公理 2 ~6 所生成的集合构成了一个全域(universe)，其中每个集合都是有穷的，并且该全域含有“无限多”的有穷集合，但没有无穷集合。公理 1 ~6 在这个全域中成立，其中 $=$，$\in$ 像通常那样解释。对于这个全域，习题 1 中作了更详细的定义和研究，并且在第十节中我们还给出了更一般的定义形式。

本节概述公理 1 ~6 的另一个模型，其中每一个“集合”只有有穷多个“元素”。(公理1 ~6 的模型是指一个集合 A 和 A 上的二元关系 $\in_A$，当把“集合”换成“A 的元素”，把 $\in$ 换成 $\in_A$ 时，公理 1 ~6 成立。第二章的第一节到第四节给出了理论模型的一般形式化定义，第三章的第五节给出了像这里带等号的模型的定义。)

阿克曼模型

在阿克曼模型(Ackermann model)中，自然数就是上面所说的“集合”，并且有一个算术关系 $\in_A$，用来表示隶属关系 $\in$。令 $\mathcal{N}=\{0, 1, 2, \cdots\}$ 为自然数的集合。定义 ω 上的“隶属关系” $\in_A$ 如下：

$x\in_A y$，当且仅当 x 是 y 的以 2 为底的幂级数展开式中的一个指数。

例如，$7=2^0+2^1+2^2$，因此 $0, 1, 2\in_A 7$，而 0 对应于空集，因为 0 的以 2 为底的幂级数的展开式中没有任何项。$4=2^2$，因而 4 的所有子集的集合 $\mathcal{P}(4)$ 是 $\{0, 4\}$，这个集合恰好由元素 0 和 4 组成。换句话说，$\mathcal{P}(4)=2^0+2^4=17$。有了这个隶属关系的解释，外延性公理对应着数论中的一个事实，即每个自然数都能仅以一种方式写成递增的以 2 为底的幂级数(习题2)。我们验证公理 3(无序对)并把其余公理的验证留作习题 3(a)。

命题 4.1 *在阿克曼模型中无序对公理成立。*

证明 为证明对每个 x 和 y，存在一个“集合” $\{x, y\}$，它仅有元素 x 和 y，需要证明：对自然数 x 和 y，存在一个自然数 z，满足在 z 的 2 的幂级数展开式中出现的指数恰是 x 和 y。如果 $x=y$，那么所需的 z 就为 2^x。如果 $x\neq y$，那么所需的 z 为 2^x+2^y。(注意，我们间接使用了一个事实，即这样的分解是唯一的(习题2)。) □

所有这些表明：为了保证“无穷”集合(例如自然数的集合)的存在，我们需要一条新的公理。一百年前，对于是否确实需要一条单独的公理存在着争议。这样的公理首先是在 Zermelo[1908, 2.3]中提出来的。以阿克曼模型为例，一条独立的“无穷公理”确实是必需的。

定义 4.2 *对任何集合 x，x 的*__后继__*(集合) $s(x)$ 是 $x\cup\{x\}$。*

公理 7(无穷) *存在一个集合 A，它包含空集并且满足只要它包含集合 x，它就包含 $s(x)$。*

注意到冯·诺伊曼(von Neumann)(作为第八节中描述的关于序数的更一般理论的一部分)给出了一个自然数的集合论式的定义(表示)，其中 $0=\varnothing$，$1=\{\varnothing\}=\{0\}=s(0)$，并且一般地，$n+1=s(n)=\{0, \cdots, n\}$。(下面会解释如何用这些集合构成一个结构，使之与带有常规算术运算和关系的自然数集同构。)通过简单的归纳可知，任何满足无穷公理条件的集合 A，都包含冯·诺伊曼的自然数集。然而，由无穷公理断言存在的集合 A 并不唯一。我们需要“削减”公理所提供的集合 A，以便恰好得到冯·诺伊曼的自然数集。

定义 4.3 *一个集合 B 是*__归纳的__(inductive)，*如果 B 包含空集 $\varnothing$，并且对每个 $y\in B$，有 $s(y)\in B$。*

因此，公理 7 断言了某些归纳集存在。按照冯·诺伊曼的做法，把自然数集合 ω 定义

为最小的归纳集。需要证明这个集合存在且唯一。

定理 4.4 *存在唯一的最小的归纳集 ω，即一个集合 ω，它自身是归纳的而且包含在每个归纳集中。*

证明 令 A 是满足公理 7 条件的任何集合，即任意的归纳集。令 $Q(x)$ 表示性质：x 是 A 的每个归纳子集 B 的一个元素。现在运用子集构造公理定义 ω 为 $\{x\in A \mid Q(x)\}$。首先验证 ω 是归纳的。如果 $x\in\omega$，那么 $x\in A$，并且对每个 A 的归纳子集 B，确有 $x\in B$。由于 A 是归纳的，由定义，$s(x)\in A$。类似地，对每个 A 的归纳子集 B，有 $s(x)\in B$。因此，$s(x)\in\omega$。

然后，假设 C 是归纳的。必须验证 $\omega\subseteq C$。由 ω 的定义，只需要证明 $A\cap C$ 是归纳的(因为它显然是 A 的子集)。为证明这一点，考虑任意的 $x\in A\cap C$。因为 $x\in A$ 且 A 是归纳的，所以 $s(x)\in A$。类似地，$s(x)\in C$。因此，$s(x)\in A\cap C$。

最后，外延性公理保证了存在唯一的归纳集包含于每个 A 的归纳子集中。如果 $B\subseteq A$ 是归纳的，那么由 ω 的定义，$\omega\subseteq B$。另一方面，如果 B 包含在每个 A 的归纳子集中，那么 $B\subseteq\omega$。因此，$B=\omega$。 □

现在有了自然数集$\mathcal{N}$的集合论定义，即 ω，并且有 ω 上的后继函数 $s(x)$，下面将证明它拥有我们所熟悉的自然数的结构$\mathcal{N}$的所有标准性质。首先证明 ω 及其后继函数满足关于自然数的每条公理，其中包括著名的自然数的“归纳公理”，这是由意大利数学家皮亚诺提出的。

定理 4.5(关于 ω 的皮亚诺公理)

(1) $0=\varnothing$ 是 ω 的元素。

(2) ω 中每个 x 在 ω 中都有一个后继 $s(x)$。

(3) ω 中不存在 x，满足 $s(x)=0$。

(4) 对 $x,y\in\omega$，$s(x)=s(y)$ 蕴涵着 $x=y$。

*(5) 假设 $Q\subseteq\omega$。令 $\varnothing\in Q$ 且只要 $x\in Q$ 就有 $s(x)\in Q$。那么 $Q=\omega$。(这条关于 ω 的断言称为**归纳公理**。)*

证明 利用这样的事实：ω 是最小的归纳集。

(1) 由定义，$\varnothing$ 在所有的归纳集中，因此，$\varnothing$ 在 ω 中。

(2) 如果 $x\in\omega$，由于 ω 是归纳的，那么 $s(x)\in\omega$。

(3) 对每个 $x\in\omega$，有 $s(x)=x\cup\{x\}$。所以 x 是 $s(x)$ 的一个元素，同时 $\varnothing=0$ 没有任何元素。因此，由外延性公理，$s(x)$ 和 x 是不同的集合。

(4) 假设 $Q\subseteq\omega$。令 $\varnothing\in Q$ 且只要 $x\in Q$，就有 $s(x)\in Q$。那么由定义，Q 本身是一个归纳集，因而必然包含最小的归纳集 ω。由外延性公理，$Q=\omega$。

(5) 假设 $x,y\in\omega$ 且 $s(x)=s(y)$。那么 $x\cup\{x\}=y\cup\{y\}$。既然 x 是这个等式左边集合的元素，那么它也是右边集合的元素，即 $x\in y$ 或 $x=y$。由对称性，$y\in x$ 或 $y=x$。因而，有 $x=y$，否则就有 $x\in y$ 且 $y\in x$。所以，只需要证明不存在包含一个元素 y 的 $x\in\omega$，满足 $x\in y$。利用性质(5)，通过归纳证明这一点。

令 $Q(x)$ 表示性质：不存在 $y\in x$，满足 $x\in y$。由于空集没有任何元素，显然有 $Q(\varnothing)$。其次，假设 $x\in\omega$ 且 $Q(x)$ 成立。需要证明 $Q(s(x))$ 也成立，即不存在 $y\in x\cup\{x\}$，满足 $x\cup\{x\}\in y$。如果存在这样一个 y，由第一个隶属关系，有 $y\in x$ 或 $y=x$。由第二个隶属关系，有 $x\in y$。因而，不管在哪种情形下，都与 $Q(x)$ 成立的假设相矛盾(在 $y\in x$ 的第一种情形中直接可得，在第二种情形中注意到，$x\in x$ 也与 $Q(x)$ 矛盾)。

因而，ω 及其后继函数满足所有的皮亚诺公理。 □

评注 4.6 注意，根据子集构造公理，ω 的性质(5)(归纳)可以等价地表述为：

(5′)假设 $Q(x)$ 是集合的一个性质。假设 $Q(\varnothing)$ 且只要 $Q(x)$ 为真，就有 $Q(s(x))$。那么对所有 $x\in\omega$，$Q(x)$ 都成立。

现在，如在定理 4.11 中所证明的，这些皮亚诺公理在同构的意义上刻画了自然数集。在证明这个结果之前，先来看看如何在 ω 上轻松地定义出我们所熟悉的自然数上的关系并且证明它们有我们想要的性质。例如，自然数上通常的序 $<$ 由隶属关系给出：对 $x, y\in\omega$，我们说 $x<y$，当且仅当 $x\in y$。

定理 4.7

(i) $\langle\omega, \in\rangle$ 是一个线性序，其中 $\varnothing$ 是最小的元素，即对每个 $x\in\omega$，有 $\varnothing\in x$ 或者 $x=\varnothing$。

*(ii) 对每个 $x\in\omega$，在此序下，$s(x)=x\cup\{x\}$ 是 x 的**直接后继**，即 $x\in s(x)$ 且不存在 $y\in\omega$，满足 $x\in y$ 和 $y\in s(x)$。*

*(iii) 每个非 0 的 $x\in\omega$ 都有一个**直接前驱**(immediate predecessor)，即某个 $y\in\omega$，满足 $s(y)=x$。*

证明

(i)为证明 $\langle\omega, \in\rangle$ 是线性序，需要证明 $\in$ 在 ω 上是反自反的和传递的，并且对每个 $x\in\omega$ 和 $y\in\omega$，或者 $x\in y$，或者 $x=y$，或者 $y\in x$。反自反性($x\notin x$)已经在定理 4.5 性质(4)的证明中确认了。

对于传递性，令 $Q(x)$ 是性质：每个 x 的元素 y 的元素 z 是 x 的元素，然后通过归纳证明：对每个 $x\in\omega$，$Q(x)$ 成立。因为空集没有任何元素 y，$Q(\varnothing)$ 当然成立。现在假设 $Q(x)$ 成立。需要证明 $Q(s(x))$ 也成立。考虑任何 z 和 y，满足 $z\in y\in s(x)=x\cup\{x\}$。由于 $y\in x\cup\{x\}$，则有 $y\in x$ 或者 $y=x$。在第一种情况下，根据归纳假设 $Q(x)$ 成立，证明完成。在第二种情况下，有 $z\in x$ 且 $x\subseteq s(x)$，所以也有 $z\in s(x)$。

对于三分法，首先通过对 y 归纳证明一个辅助结果：对每个 $x\in\omega$，如果 $x\in y$，那么 $y=s(x)$ 或者 $s(x)\in y$。如果 $y=\varnothing$，这个结果是显然的，因为任何集合都不是 $\varnothing$ 的元素。其次，考虑 $s(y)$，假设 $x\in s(y)$。由 $s(y)$ 的定义，$x=y$ 或者 $x\in y$。在第一种情况下，$s(y)=s(x)$ 已为所需。在第二种情况下，由归纳，$y=s(x)$ 或者 $s(x)\in y$，所以 $s(x)\in s(y)$。现在通过对 x 归纳证明，对每个 $y\in\omega$，$x\in y$，$x=y$ 或者 $y\in x$(三分法)。对 $x=\varnothing$，这可通过对 y 归纳立即得到。最后，考虑 $s(x)$。由归纳，有 $x\in y$，$x=y$ 或者 $y\in x$。在第一种情况下，我们的辅助结果说明 $y=s(x)$ 或者 $s(x)\in y$ 已为所需。在其他情况下，所需结论由 s 的定义得到。

当然，由于任何集合 x 都不是 $\varnothing$ 的元素，$\varnothing$ 是线性序 $\langle\omega, \in\rangle$ 的最小元素(由三分法知)。

(ii)下面证明 $s(x)$ 是 x 的直接后继。由 $s(x)$ 的定义立即可得，$x\in s(x)$。如果存在任何 $y\in\omega$，满足 $x\in y\in s(x)$，那么(由第二个隶属关系的假设) $y=x$ 或 $y\in x$。不论哪种情况，第一个假设($x\in y$)和上面建立的 ω 上 $\in$ 的反自反性矛盾。

(iii)考虑性质 $Q(x)$：$x=\varnothing$ 或存在唯一的 $y\in\omega$，满足 $s(y)=x$。显然 $Q(\varnothing)$ 成立，并且由皮亚诺第四条公理对每个 $x\in\omega$，$Q(s(x))$ 成立。因此，根据归纳，$Q(x)$ 对每个 $x\in\omega$ 成立。

这就完成了三个事实的证明。 □

从现在开始，特别是把 ω 当作自然数集时，经常把 ω 上的隶属关系写成 $<$，空集写成 0

且它的直接后继 $s(0)$ 是 1。用这样的表达方式，可以给出归纳原理的另一个常见的形式：

(5″)如果 $Q(x)$ 是集合的性质且存在一个 $x\in\omega$，满足 $Q(x)$，那么就存在一个最小的 $x\in\omega$，满足 $Q(x)$，即存在 $x\in\omega$，满足 $Q(x)$，但不存在 $y<x$（即 $y\in x$）使 $Q(y)$ 成立。

这个版本的归纳公理（比如说根据(5′)得到）的证明是习题 4。

然而要确定通常的加法和乘法等数论函数，除了到目前为止所叙述的归纳原理之外，还需要更多的东西。我们所欠缺的是关于（函数的）*归纳定义*(definition by induction)或*递归定义*(definition by recursion)的想法。一个自然数集上的函数 f 的归纳（或递归）定义是这样处理的：确定 f 在 0 处的值并且提供一个规则，使其对每个 $x>0$，以 f 在小于 x 的数上的取值来决定 $f(x)$ 的值。一个典型的例子是用后继定义的加法。我们希望在 ω 上定义二元函数 $x+y$，通过对每个 $y\in\omega$，用归纳定义函数 $f(x)=x+y$，按定义 $f(0)=0+y$ 就是 y。对归纳的情形，令 $f(s(x))=s(f(x))=s(x+y)$。这样在集合论中发展 ω 上算术的关键问题是如何证明存在一个函数（$\omega\times\omega$ 的一个子集就是一个函数）满足这样的定义。首先引入序列这个重要的概念。在证明关于递归定义函数的一个相当通用的定理时，它是一个有用的工具，使得能由这样的递归推导出定义在自然数上的许多函数的存在性。（第九节中给出了函数递归定义的更一般的形式。）

定义 4.8 *一个（B 中的或 B 的元素的）n-***序列***是一个定义域为 n，值域包含在 B 中的函数 σ。一个 n-序列的长度是 n。一个（B 中的）ω-***序列***是一个定义域为 ω，值域包含在 B 中的函数 σ。如果 $B=2=\{0,1\}$，称这样的函数为***二元序列**。如果 σ 是一个 n-序列，τ 是一个 m-序列，那么 σ 和 τ 的***毗连** $\sigma^\frown\tau$，是一个 $(n+m)$-序列 ρ，定义为：对 $i<n$，$\rho(i)=\sigma(i)$；对 $j<m$，$\rho(n+j)=\tau(j)$。我们常常称 $\sigma(i)$ 为序列 σ 的第 $s(i)$（即 $i+1$）个元素。因而，序列 σ 的第一个元素是 $\sigma(0)$。

一个（B 中的）n-序列是（B 中的）长度为 n 的元素串的形式化概念。σ 和 τ 的毗连对应于把串 τ 放在串 σ 的后面。注意到，一个长度为 0 的序列是一个定义域为 $\varnothing$ 的函数，且它自身是一个没有任何元素的函数，即仅仅是空集 $\varnothing$。

现在证明一个通过递归定义的函数的存在性定理，其中递归对应于在可计算函数定义中使用的原始递归定义的形式。（参见第三章第八节，特别是其中的习题。）

定理 4.9（递归或归纳定义） *如果 j: $A\to B$ 和 h: $\omega\times B\times A\to B$ 是任意的映射，那么存在唯一的映射 f: $\omega\times A\to B$，满足对每个 $x\in A$ 和 $n\in\omega$，有 $f(0,x)=j(x)$ 和 $f(s(n),x)=h(n,f(n,x),x)$。*

证明 令 C 表示从 A 到 B 的函数的集合。令 $T=\{\sigma\mid$ 对某个 $n\in\omega$，σ 是 C 中一个 $s(n)$-序列，使得对每个 $x\in A$ 和 $i<n$，有 $\sigma(0)(x)=j(x)$ 且 $\sigma(s(i))(x)=h(i,\sigma(i)(x),x)\}$。（此处的想法是：如果 $\sigma\in T$，那么序列 σ 的第一个元素是函数 j 且如果 g 和 k 分别是序列的第 i 个和第 $i+1$ 个元素，那么 $k(x)=h(i,g(x),x)$。因此，对这样一个序列 σ，$\sigma(s(i))(x)$ 看起来像所要求的函数 f 在 $\langle s(i),x\rangle$ 处的值。）（习题 5 表明应用集合构造公理可以证明集合 T 是存在的。）

首先归纳证明：对每个 $n\in\omega$，T 中存在一个长度为 $s(n)$ 的序列。对 $n=0$，注意到仅含一个元素 j 的序列 σ 在 T 中（$\sigma(0)=j$ 且 $\mathrm{dom}\sigma=1$）。假设长度为 $s(n)$ 的 $\sigma\in T$，那么序列 $\sigma^\frown g$ 的长度为 $s(s(n))$，其中 $g(x)=h(n,\sigma(n)(x),x)$。正如我们所需要的它也在 T 中。

然后归纳证明：如果 $\sigma,\tau\in T$，那么 σ 和 τ 是相容的(compatible)，即如果 $n\in\mathrm{dom}\sigma\cap\mathrm{dom}\tau$，

那么 $\sigma(n)=\tau(n)$。对 $n=0$，只要注意到对每个 $\sigma\in T$，根据定义，有 $\sigma(0)=j$。对于归纳步，假设给出的性质对 n 成立，σ，$\tau\in T$ 且 $s(n)\in \mathrm{dom}\sigma\cap\mathrm{dom}\tau$。由归纳假设 $\sigma(n)=\tau(n)$。由 T 的定义，对每个 $x\in A$，有 $\sigma(s(n))(x)=h(n,\ \sigma(n)(x),\ x)=h(n,\ \tau(n)(x),\ x)=\tau(s(n))(x)$，因此，$\sigma(s(n))=\tau(s(n))$。

现在可以在 $\omega\times A$ 上定义所需的函数 f 为 $\{\langle\langle n,\ x\rangle,\ y\rangle\in\omega\times A\times B\mid$ 存在一个 $\sigma\in T$，满足 $n\in\mathrm{dom}\sigma$ 且 $\sigma(n)(x)=y\}$。根据刚才已经证明的第二个事实可知，这个集合是一个函数，同时由第一个事实可知，它的定义域为 $\omega\times A$。由 T 的定义及其元素的相容性，这个函数满足定理对 f 所要求的性质。

为说明定理中描述的映射 f: $\omega\times A\to B$ 的唯一性，假设 f 和 f' 都满足对 f 的要求。归纳证明：对每个 $x\in A$，$f(n,\ x)=f'(n,\ x)$。对 $n=0$，这就是我们的假设，即对每个 $x\in A$，有 $f(0,\ x)=j(x)=f'(0,\ x)$。对 $s(n)$，注意到，由我们对 f 和 f' 的选择，$f(s(n),\ x)=h(n,\ f(n,\ x),\ x)$ 且 $f'(s(n),\ x)=h(n,\ f'(n,\ x),\ x)$，同时，由归纳，$f(n,\ x)=f'(n,\ x)$，因此，$f(s(n),\ x)=f'(s(n),\ x)$。 □

现在应用这个定理说明加法（+）、乘法（·）和指数（*）在 ω 上都能由通常的递归方式定义。

定理 4.10 存在 ω 上（唯一）的二元函数 +，·和 *，对每个 $x\in\omega$ 满足下面的递归条件：

(i) $0+x=x$ 且 $s(n)+x=s(n+x)$

(ii) $0\cdot x=0$ 且 $s(n)\cdot x=n\cdot x+x$

(iii) $x*0=1$ 且 $x*s(n)=(x*n)\cdot x$

证明

(i) 如果定义定理中的映射 h: $\omega\times\omega\times\omega\to\omega$ 为 $h(n,\ x,\ y)=s(x)$ 且令 $j(x)=x$，那么映射 f: $\omega\times\omega\to\omega$ 存在且对加法满足 (i) 中条件。

(ii) 如果定义映射 h: $\omega\times\omega\times\omega\to\omega$ 为 $h(n,\ x,\ y)=x+y$ 且令 $j(x)=0$，那么由定理得到的函数 f 对乘法满足 (ii) 中的条件。

(iii) 如果定义映射 h: $\omega\times\omega\times\omega\to\omega$ 为 $h(n,\ x,\ y)=x\cdot y$ 且令 $j(x)=1$，再令 f 为由定理得到的函数，那么 $g(x,\ y)=f(y,\ x)$ 对指数运算满足 (iii) 中的条件。 □

正如在 ω 上已经定义的那样，现在可以验证，自然数上的常规算术法则对 <，+ 和 · 成立。更多的函数，包括第三章第八节中所谓的“可计算”函数，都能应用定理 4.9 证明其存在性。特别地，令 A 为 ω^n 且 B 为 ω，可以定义有 $n+1$ 个变量的函数。（见习题 8。）现在还可以很容易地定义集合论形式的整数、有理数和实数以及相应的 <，+ 和 ·。（见习题 9 ~ 11。）

最后，证明定理 4.5 中给出的皮亚诺后继公理在同构意义下决定了带后继的自然数结构。

定理 4.11 如果 f: $A\to A$ 和 g: $B\to B$ 都是满足皮亚诺后继公理的映射（定理 4.5 中列出了在 ω 上的后继映射 s 的性质 (1) ~ (5)），那么存在一个双射 h: $A\to B$，满足对每个 $x\in A$，有 $h(f(x))=g(h(x))$。（这样的双射称为结构 $\langle A,\ f\rangle$ 与 $\langle B,\ g\rangle$ 的一个**同构**（isomorphism）。）

证明 首先，注意到只需要证明当 $\langle A,\ f\rangle=\langle\omega,\ s\rangle$ 时，定理成立，因为给定任何 $\langle A,\ f\rangle$ 与 $\langle B,\ g\rangle$，根据定理在第一个结构为 $\langle\omega,\ s\rangle$ 时的特殊情况，可以复合双射 h: $\omega\to A$ 和 k:

$\omega \to B$，得到想要的双射 kh^{-1}：$A \to B$。(回忆 $kh^{-1}(x) = k(y)$，其中 $h(y) = x$。)由第三节习题8可知，kh^{-1}是一个双射。这个双射是一个同构，是因为 $kh^{-1}(f(x)) = k(y)$，其中 $h(y) = f(x)$。由对 h 的选择，$h(s(z)) = f(x)$，其中 $h(z) = x$。因此，$kh^{-1}(f(x)) = kh^{-1}(h(s(z))) = k(s(z)) = g(k(z))$(由 k 的选择) $= g(kh^{-1}(x))$(因为 $z = h^{-1}(x)$)。

于是，假设给定的 g：$B \to B$ 满足皮亚诺后继公理。希望证明存在所需的同构 h：$\omega \to B$。由第一条皮亚诺公理，$\varnothing \in B$。现在可以利用定理4.9证明存在一个映射f：$\omega \times \omega \to B$，满足对每个 x，$n \in \omega$，有 $f(0, x) = \varnothing$ 且 $f(s(n), x) = g(f(n, x))$。(在定理中对 n，$x \in \omega$ 和 $y \in B$,令 $j(x) = \varnothing$ 且 $h(n, y, x) = g(y)$。)现在断言所需的同构由 $h(n) = f(n, 0)$ 给出。当然，由 f 的选择，h 的定义域是 ω 且对每个 $n \in \omega$，有 $h(s(n)) = g(h(n))$。因而，只需要验证 h 是 1-1 和映满的。

为说明 h 是 1-1 的，对 n 归纳证明不存在 $m > n$，满足 $h(n) = h(m)$。对 $n = 0$，$h(n) = \varnothing$ 且如果 $m > 0$，那么由定理 4.7(iii)对某个 $y \in \omega$，满足 $m = s(y)$。因而 $h(m) = h(s(y)) = g(h(m))$。但由关于 g 的皮亚诺第三公理，对任何 m 都有 $g(h(m)) \neq \varnothing$。然后，考虑 $s(n)$。由于 $m > s(n)$，再次有 $m = s(y)$ 对某个 y 成立。如果是这样，则有 $g(h(n)) = h(s(n)) = h(s(y)) = g(h(y))$，那么由关于 g 的皮亚诺第四公理，有 $h(n) = h(y)$。由于这将与对 n 的归纳假设矛盾，所以完成了归纳且证明了 h 是 1-1 的。(注意，$s(y) > s(n)$ 意味着 $n \cup \{n\} = s(n) \in s(y) = y \cup \{y\}$ 且 $n \neq y$。因而 $n \in y$，所以 $y > n$，这恰是应用归纳假设所需要的。)

最后，为证明 h 是映满的，通过对 B 应用归纳公理，证明存在某个 $m \in \omega$，使得对每个 $x \in B$，有 $h(m) = x$。对 $x = \varnothing$，已有 $h(0) = \varnothing$。假设 $h(m) = x$。只需要找到一个 n，满足 $h(n) = g(x)$。然而，由对 h 的选择，$h(s(m)) = g(h(m)) = g(x)$，这正是所需要的。 □

习题

1. 通过归纳定义集合 $R(n)$ 如下：$R(0) = \varnothing$ 且 $R(n+1) = \mathcal{P}(R(n))$。令 $R = \cup\{R(n) \mid n \in \mathcal{N}\}$。假设 R 存在(R 存在性的形式化证明要用到替换公理，见第六节习题1，更一般的情形见第十节)证明如下各项：
   (a)每个 $R(n)$ 都是有穷的。
   (b)对每个 $n \in \omega$，有 $R(n) \subsetneq R(n+1)$。
   (c)公理 2 ~ 6 在 R 中成立，即 $\varnothing \in R$ 且对每个 x，$y \in R$ 和性质 $P(z)$，集合 $\{x, y\}$，$\cup x$，$\{z \in x \mid P(z)\}$ 和 $\mathcal{P}(x)$ 都在 R 中。
   (d) $\cup R = R$。
   (e) R 包含在满足(c)的每个集合中。
   (f)无穷公理在 R 中不成立，即不存在 $x \in R$，满足 $\varnothing \in x$ 且对每个 $y \in x$，有 $s(y) \in x$。
2. 证明：(作为一般的算术习题)对每个 $x \in \mathcal{N}$，存在 $a_1 < \cdots < a_n$，满足 $x = 2^{a_1} + \cdots + 2^{a_n}$，并且如果 $2^{a_1} + \cdots + 2^{a_n} = 2^{b_1} + \cdots + 2^{b_m}$，其中 $b_1 < \cdots < b_m$，那么 $m = n$ 且对 $1 \leqslant i \leqslant n$，$a_i = b_i$。
3. 在阿克曼模型中，“集合”是自然数且隶属关系定义为：$x \in_A y$，当且仅当 x 是 y 的以 2 为底的幂级数展开式中的一个指数。证明在此模型中下面的结论成立：
   (a)公理(1) ~ (2)和(4) ~ (6)成立。
   (b)无穷公理不成立。
4. 根据归纳公理的版本(5′)证明版本(5″)。
5. 证明：集合 $T = \{\sigma \mid$ 对某个 $n \in \omega$，σ 是 C 中一个 $s(n)$-序列，满足对每个 $x \in A$ 和 $i < n$，有 $\sigma(0)(x) = j(x)$ 且 $\sigma(s(i))(x) = h(i, \sigma(i)(x), x)\}$ 存在。
6. (自然数算术)对所有 a，b，$c \in \omega$，归纳证明：
   (a) $(a+b)+c = a+(b+c)$(对 c 归纳)。

(b) $a+1=1+a$（对 a 归纳）。

(c) $a+b=b+a$（对 b 归纳）。

(d) $a+b=a+c \Rightarrow b=c$（对 a 归纳）。

(e) $a(b+c)=ab+ac$（对 c 归纳）。

(f) $S(b)\cdot a=ba+a$（对 a 归纳）。

(g) $(a\cdot b)\cdot c=a\cdot(b\cdot c)$（对 b 归纳）。

(h) $a<b \Rightarrow a+c<b+c$（对 c 归纳）。

(i) $a<b$ 且 $c\neq 0$ 蕴涵 $ac<bc$（对 c 归纳）。

7. 证明：阶乘函数（$n!=1\cdot 2\cdot \ldots \cdot(n-1)\cdot n$）存在。

8. 证明：对每个 n，第三章第八节的习题中定义的所有（部分）n 个变量的递归函数存在。

9. 令 $\mathcal{Z}=2\times\omega-\{\langle 0,0\rangle\}=\{\langle i,n\rangle \mid i=0$ 或 1 且 $n\in\omega\}-\{\langle 0,0\rangle\}$。在 $\mathcal{Z}$ 上定义如下的一个二元关系 $<$ 和二元运算 $+$ 及 $\cdot$：

(i) $\langle i,n\rangle<\langle j,m\rangle$，当且仅当 $i<j$ 或 $i=j=0$ 且 $n>m$ 或 $i=j=1$ 且 $n<m$。

(ii) $\langle i,n\rangle+\langle i,m\rangle=\langle i,n+m\rangle$；对 $i\neq j$，$\langle i,n\rangle+\langle j,n\rangle=\langle 1,0\rangle$；对 $i\neq j$ 且 $n\neq m$，$\langle i,n\rangle+\langle j,m\rangle=\langle k,p\rangle$，其中，如果存在 $x\in\omega$，满足 $m+x=n$，$k=i$，如果存在 $x\in\omega$，满足 $n+x=m$，$k=j$；p 在两种情况中都是所要求的 x。

(iii) $\langle i,n\rangle\cdot\langle j,m\rangle=\langle k,n\cdot m\rangle$，其中当 $i=j$ 时，$k=1$，否则 $k=0$。

证明：如果把 $\langle 0,n\rangle$ 和 $\langle 1,n\rangle$ 分别记为 $-n$ 和 $+n$，那么已经定义的整数的整合论表示，即带有 $<$，$+$ 和 $\cdot$ 的 $\mathcal{Z}$，同构于通常的整数结构。还可以直接对 $\mathcal{Z}$ 中的整数验证通常的算术律，就像在习题 6 中对 ω 和 $\mathcal{N}$ 所做的那样。

10. 令 $\mathcal{Z}^+=\{\langle 1,n\rangle \mid n\in\omega\}-\{\langle 1,0\rangle\}$。考虑笛卡儿积 $S=\mathcal{Z}\times\mathcal{Z}^+$。

(a) 证明下面给出的 S 上的关系 R 是一个等价关系：$\langle a,b\rangle R\langle c,d\rangle$ 当且仅当 $a\cdot d=b\cdot c$。

(b) 令 $\mathcal{Q}$ 是 R 的等价类的集合。类似于上面在 $\mathcal{Z}$ 上定义的 $<$，$+$ 和 $\cdot$，在 S 上定义一个二元关系 $<$ 及二元运算 $+$ 和 $\cdot$ 如下：

i) $\langle a,b\rangle<\langle c,d\rangle$ 当且仅当 $a\cdot d<c\cdot b$。

ii) $\langle a,b\rangle+\langle c,d\rangle=\langle a\cdot d+c\cdot b,\ b\cdot d\rangle$。

iii) $\langle a,b\rangle\cdot\langle c,d\rangle=\langle a\cdot c,\ b\cdot d\rangle$。

证明：$<$，$+$ 和 $\cdot$ 在 $\mathcal{Q}$ 上是良定义的，即这些运算的结果（在关于 R 等价意义上）独立于等价类中代表元的选择。

(c) 证明：对 $a,b\in\mathcal{Z}$，当记 $\langle a,b\rangle$ 为 $\frac{a}{b}$ 时，这些定义对应于有理数上通常的运算。因而，在我们的理论中，就定义了有理数（的集合论表示）。

11. 令 $\mathcal{R}=\{x\subseteq\mathcal{Q} \mid x$ 在 $\mathcal{Q}$ 中有上界但无最大元且如果 $q<r\in x$，则 $q\in x\}$。对 $x,y\in\mathcal{R}$，定义 $\mathcal{R}$ 上的 $<$ 和 $+$ 如下：

i) $x<y$，当且仅当对每个 $q\in x$，存在 $r\in y$，满足 $q<r$。

ii) $x+y=\{q+r \mid q\in x$ 且 $r\in y\}$。

(a) 证明：$\mathcal{R}$ 这个集合存在且 $<$，$+$ 分别定义了 $\mathcal{R}$ 上的二元关系和二元运算，当 $q\in\mathcal{Q}$ 等同于 $\{r\in\mathcal{Q} \mid r<q\}\in\mathcal{R}$ 时，这个关系和运算推广了 $\mathcal{Q}$ 上的相应关系和运算。

(b) 定义 $\mathcal{R}$ 上的一个二元运算 $\cdot$，它推广了 $\mathcal{Q}$ 上的相应运算（这里对于 $\mathcal{Q}$ 采用上面的等同认定）并且带有 $<$，$+$ 和 $\cdot$ 的集合论结构 $\mathcal{R}$ 同构于通常的实数系统。

第五节　替换、选择和基础

公理 1 ~ 7 实际上都包含在 Zermelo[1908，2.3] 为集合论提出的第一个公理系统中。但是，大约在 1920 年，人们发现有些相当简单的集合仅凭这些公理是无法证明其存在性的。这里有两个用非形式化表述的集合，根据这些公理无法证明它们的存在性。

(1) $\{\omega, s(\omega), s(s(\omega)), \cdots\}$

(2) $\{\omega, \mathcal{P}(\omega), \mathcal{P}(\mathcal{P}(\omega)), \cdots\}$。

需要另外的公理来证明这种集合的存在性，它就是替换公理。Fraenkel[1922a, 2.3]和Skolem[1922, 2.3]把它加进了策梅洛公理里。正如在策梅洛的子集构造公理中使用的集合性质 $P(x)$ 的(非形式化)概念一样，需要一个性质 $Q(x, y)$ 的(非形式化)概念将 x 和 y 联系起来以定义替换公理。

定义 5.1 二元对 $\langle x, y\rangle$ 的性质 $Q(x, y)$(把集合 x 关联到集合 y)称为一个**函数性质**(functional property)，如果对每个 x，$Q(x, y)$ 和 $Q(x, z)$ 蕴涵着 $y=z$。如果 Q 是一个函数性质，那么常将 $Q(x, y)$ 写成 $Q(x)=y$。

换句话说，一个函数性质必须是单值的。注意没有要求 Q 是全局的，即可能存在集合 x，对其没有 y，满足 $Q(x, y)$。

例 5.2

(i)性质 $Q(x, y)$ 成立，当且仅当 y 是 x 的子集，不是函数性质。

(ii)性质 $Q(x, y)$ 成立，当且仅当 y 是 x 的幂集，是函数性质。

公理 8(替换) 令 $Q(x, y)$ 为一个函数性质。则对任何集合 z，存在一个集合 w，满足对所有 y，$y\in w$ 当且仅当存在一个 $x\in z$，满足 $Q(x, y)$ 成立。

换句话说，一个集合在一个函数性质下的像是一个集合。根据外延性公理，公理 8 存在的集合 w 由 z 和 Q 决定。记为 $w=\{y \mid y=Q(x)$ 且 $x\in z\}$。

应用替换公理到定义域为 ω 的函数性质 $Q(x, y)$ 和 $Q'(x, y)$，可知(1)和(2)中描述的集合存在，其中

(1) $Q(0, y)$ 成立，当且仅当 $y=\omega$ 且 $Q(n, y)$ 对 $n>0$ 成立，当且仅当 y 是对 ω 取后继运算 n 次得到的结果。

(2) $Q'(0, y)$ 成立，当且仅当 $y=\omega$ 且 $Q(n, y)$ 对 $n>0$ 成立，当且仅当 y 是对 ω 取幂集运算 n 次得到的结果。

评注 5.3 函数性质的概念比通常把函数视为有序对的单值集合的数学定义更一般。当然，替换公理中所使用的函数性质的概念和子集构造公理中所使用的性质的概念同样模糊。它们都不曾导致任何悖论。不过，与对子集构造公理的处理一样，现代集合论可以精确地表述这个公理为：只允许那些可以用基于 $\in$ 和 $=$ (以集合为参数)的语言中的谓词逻辑公式表示的函数性质。(见第六节)。

还要注意到子集构造公理实际上是替换公理的一种特殊情况。为说明这一点，给定集合的性质 $P(x)$，定义集合的函数性质 $Q(x, y)$ 为：$Q(x, y)$ 当且仅当 $P(x)$ 且 $x=y$。那么根据替换公理，子集构造公理断言存在的集合 $B=\{x\in A \mid P(x)\}$ 是 A 在 Q 下的像。

另一方面，替换公理不是公理 1~7 的推论。可以通过构造一个公理 1~7 的模型，且它不是公理 8 的模型来说明这点。(直觉上，依赖于这样的想法，即如果一些公理(在任何集合论模型中)为真，那么这些公理的推论(在任何集合论模型中)也为真。这是一种熟悉的论证：如果有一个群 G，(在包含常元符号 e 作为单位元和乘法运算符 · 的群的语言中的)某个语句 ϕ 是错的，那么 ϕ 就不是群论公理的推论(定理)。这些概念在第二章和第三章的第五节所研究的谓词逻辑中，特别是在第二章的定理 7.2 和第三章的第五节习题 1 关于有效性的结果中描述得更精确，所需要的模型就是一些带有常规的隶属关系的集合类。下面将简要地

介绍公理 1 ~ 7 的这个模型的构造，并指出为什么它不满足替换公理。

定理 5.4 存在公理 1 ~ 7 的一个模型，但它不是公理 8（替换公理）的模型。

证明 令 $S(0)=\omega$ 且对 $n\in\omega$ 归纳定义 $S(n)$ 为 $S(n+1)=\mathcal{P}(S(n))$。最后，令 $S=\bigcup\{S(n)\mid n\in\omega\}$。现在可以验证带有通常的隶属关系 $\in$ 的 S 满足公理 1 ~ 7。（注意，需要替换公理来验证 S 存在。$\{S(n)\mid n\in\omega\}$ 就是上面(2)中描述的集合，公理 8 证明了它的存在性。）另一方面，S 不满足替换公理。实际上，函数关系 $Q(x,y)$ 定义为（$x\in\omega$ 且 $y=S(x)$）提供了我们需要的反例，因为 $\{S(n)\mid n\in\omega\}$ 不是 S 的元素。细节的验证留作习题 1。 □

康托（Cantor）在自己的工作中使用了一个类似于替换公理的原理：集合 A 在 1-1 的函数性质下的像 B 是一个集合。这个原理是他的基数理论的基础。（见第七节和第十一节。）那么这个原理等价于替换公理吗？事实证明，如果承认康托的另一个基本假设：康托良序原理，那么取 A 在函数性质 Q 下的像 B，这种最常见的情形可以相当容易地归约为取 A 在一个相应的 1-1 的函数性质 Q' 下的像 B。在陈述这个原理前，需要一些重要的定义。

定义 5.5 假设 $(A,<)$ 是一个偏序，$a\in A$ 且 $B\subseteq A$。

(i) a 是 B 的**下界**（lower bound），如果对所有 $b\in B$，有 $a\leqslant b$；a 是 B 的**上界**（upper bound），如果对所有 $b\in B$，有 $a\geqslant b$。

(ii) c 是 B 的**最大下界**（greatest lower bound，glb），如果 c 是 B 的下界且是 B 的所有下界组成的集合的上界，即若 x 是 B 的任一下界，则 $x\leqslant c$。类似地，c 是 B 的**最小上界**（least upper bound，lub），如果 c 是 B 的上界且是 B 的所有上界组成的集合的下界，即若 x 是 B 的任一上界，则 $c\leqslant x$。

(iii) c 是 B 的**最小元**（least element），如果 c 是 B 的元素且为 B 的下界。或者说，它是 B 的元素且为 B 的最大下界。类似地，c 是 B 的**最大元**（greatest element），如果 c 是 B 的元素且为 B 的上界。

注意 B 最多能有一个最小元：如果 c 和 d 是 B 的最小元，那么 $c\leqslant d$ 且 $d\leqslant c$，因而 $c=d$。类似地，B 最多能有一个最大元。

定义 5.6 一个线性序 $\langle A,<\rangle$ 是**良序的**，如果 A 的每个非空子集 B 都有最小元。（在这种情况下，也称 $<$ **良序化**（well order）A 或是 A 的**良序**（well ordering）。）一个偏序 $\langle T,<_T\rangle$ 是一棵**树**，如果它有最小元，并且对每个 $x\in T$，x 的前驱的集合 $\{y\mid y<_T x\}$ 被 $<_T$ 良序化。

例 5.7

(i) $\langle\omega,<\rangle$ 是良序的。这个事实可由上文描述的归纳原理的版本(5″)立即得到。

(ii) 以扩张（若 $\mathrm{dom}\sigma\subset\mathrm{dom}\tau$，且对所有的 $x\in\mathrm{dom}\sigma$，有 $\sigma(x)=\tau(x)$，则 $\sigma<\tau$）为序的有穷二元序列的集合是一棵树。

(iii) 令 Q^+ 表示正有理数。那么 $\langle Q^+,<\rangle$ 是一个线性序但它不是良序的，因为它没有最小元。

(iv) 实数的闭区间 $[1,2]$ 加上实数上常用的序是一个线性序但不是良序。这个区间中平方大于 2 的有理数集合没有最小元，因为 $\sqrt{2}$ 是无理数。

事实上，在最后两个例子中集合上的自然序不是良序并不意味着对这些集合不存在良序。构造关于 Q^+ 的良序留给读者作为习题。（见习题 4。）但构造实数的良序却不是一个习题，尽管后面的定理 5.9 基于额外的公理说明每个集合都能被良序化。每个集合可良序化被

康托所认同和使用，并以康托良序原理著称。策梅洛曾尝试证明它，并且在其尝试中第一次引入了一个看似可靠的集合论公理，这个公理却蕴涵康托良序原理。

公理9(选择) 如果A是非空集合的集合，那么存在一个定义域为A的函数f，满足对每个$x\in A$，$f(x)\in x$。这个函数f称为A的一个**选择函数**(choice function)。

一个选择函数恰好从A的每个元素中选出一个代表元。如果A是有穷的，那么这不存在任何令人惊讶或模糊的地方。事实上，关于有穷集合A的选择函数的存在性，根据通常的逻辑法则可以得到。这个公理的真正内容是说：集合A是由无穷多个集合y构成的，人们可以同时从每个y中选出一个元素。康托和策梅洛认为这是显然的。在康托的情形中，他有一个看似集合的概念，其中所有集合以一种确定的(超穷)序被“创造”出来。在这个序下，新集合的元素产生于该集合之前。这样的话，他当然就可以通过让函数f从每个$x\in A$中选出在集合的这种主序下出现的第一个元素来验证选择原理。这个步骤本质上是从良序原理导出了选择公理。策梅洛的做法则相反，在定理10.2中将证明策梅洛的结论。

伯特兰·罗素(Bertrand Russell)对此作了注解，如果一个人有无穷多双靴子，那么他就能轻易地定义一个选择函数：例如从每双中选出左脚的靴子。另一方面，如果给我们无穷多双无法分辨的袜子，或许就不能制定这样的规则。对于康托的观点有一个隐喻的说法是这样的。如果袜子在工厂里是按顺序生产的，那么就取出每双袜子中第一只生产的。早期关于选择公理的许多讨论就是以这样的语言阐述的。选择公理在数学中以许多不同但等价的形式被广泛运用。下面简要地提及其中一些。

定义5.8 令$\langle A, <\rangle$是一个偏序。A的一个子集B称为**一条链**(chain)，如果对任何$b_1, b_2\in B$，或者$b_1\leqslant b_2$，或者$b_2\leqslant b_1$。A的一个元素x称为(在序$<$下的)一个**极大**(maximal)**元**，如果对任何$y\in A$都没有$x<y$。A的一个元素x称为(在序$<$下的)一个**极小**(minimal)**元**，如果对任何$y\in A$都没有$y<x$。

定理5.9 下面的断言是等价的：

(i)对每个由非空集合组成的集合A，存在一个定义域为A的函数f，满足对每个$x\in A$，$f(x)\in x$。

(ii)对每个集合A，存在一个关系$R\subseteq A\times A$良序化A。

(iii)令$\langle A, <\rangle$为任何非空偏序。如果每条链$B\subseteq A$都有上界，那么A就有一个极大元。

(i)是选择公理。(ii)是康托的良序原理。(iii)是佐恩引理(Zorn's lemma)，在研究生分析和代数课程中，它是这个公理最常用的形式。

可以(甚至经常)给出这些等价的证明，事实上，Zermelo[1904，2.3]从选择公理证明良序原理要先于现在的由冯·诺伊曼发现的良序化方法。尽管如此，还是延后此证明，直到有工具能给出直观清晰的证明。这个工具就是超穷序数理论和在序数上用超穷归纳定义函数的理论。有了这些工具，证明只需要对康托原来对于良序原理所作的验证作些许的改进。

选择公理一直以来是集合论公理中最有争议的，人们一直都不知道它和其他的公理是相容的。直到哥德尔(Gödel)[1938，1939，2.3]证明了：如果存在一个公理1~8的模型，那么其中就存在一个模型，康托的直觉在其中是对的，因为此时存在一个具有可定义良序的模型，所有集合都是按这个序产生的。当然，在这个模型中，良序原理和选择公理是成立的。另一方面，Cohen[1963，1964，3.3]证明了存在公理1~8的一些模型，其中选择公理是错

误的。即使是这些结果的描述也已经超出了我们讲述的范围。这些构造和证明在 Cohen [1966, 3.3]或大多数现行的研究生公理集合论的教材中都能找到，例如 Kunen[1980, 3.3]。现在，大多数数学家都自由地使用选择公理，但是有时也明确地注明他们的结果用到了选择公理。

在本书的最后，给出一个技术性的公理：基础(foundation)公理或正则(regularity)公理。这个公理由弗兰克尔(Fraenkel)提出，他通过把集合论限制在某些集合上来排除各种可能的反常情况，其中这些集合的元素必须先于集合本身被“知道”或被“构造”。因此，举例来说，将不存在集合$\{x, y, z\}$，满足$x\in y\in z\in x$。更一般地，也不存在无穷递减的$\in$-序列，即ω-序列σ对所有$n\in\omega$，满足$\sigma(n+1)\in\sigma(n)$。

公理 10(基础) *每个非空集合A都有一个元素y，与A不相交(即$y\cap A=\varnothing$)。*

可以将此公理中假定的A的这个元素y描述为A的一个$\in$-极小元($\in$-minimal element)，即$y\in A$但没有$x\in y$是属于A的。这个公理不仅排除了满足$x\in x$的集合x的存在性，还排除了所有无穷递减$\in$-序列。

定理 5.10 *不存在ω-序列σ，对所有$i\in\omega$，满足$\sigma(i+1)\in\sigma(i)$。*

证明 假设存在这样一个ω-序列σ。令$A=\{\sigma(n)\mid n\in\omega\}$为$\sigma$的值域。应用基础公理得到$A$的一个$\in$-极小元$y$，满足对某个$n$，有$y=\sigma(n)$。但由$\sigma$的选择，$\sigma(n+1)\in\sigma(n)$，且由$A$的定义，$\sigma(n+1)\in A$，因而$\sigma(n)$不是$\in$-极小元，得到矛盾。 □

特别地，基础公理排除了$x\in x$这种不合理的集合x。定理 5.10 的逆命题需要选择公理。

定理 5.11 *基础公理可以由公理 1～9 和不存在无穷递减$\in$-序列的假设得到。*

证明 假设A是一个非空集合，且没有元素y，满足$y\cap A=\varnothing$，特别地，$\varnothing\notin A$。令k是A的一个选择函数，即对每个$y\in A$，$k(y)\in y$。令$B=A$，对任何固定的$a\in A$，$j(x)=a$且$h(n, x, y)=k(x)$，应用定理 4.9，得到函数f: $\omega\times A\to A$，对每个$n\in\omega$，满足$f(n+1, x)\in f(n, x)\in A$。现在若令$\sigma(n)=f(n, a)$，则$\sigma$显然是一个无穷递减的$\in$-序列，得到矛盾。 □

基础公理实际上是简约的结果。它仅排除了通常在数学中不使用的集合。这个观点可以解释得更加确切。一旦发展了序数的理论，就可以证明，如果存在公理 1～8 或公理 1～9 的模型，那么也存在一个满足基础公理的模型。这个模型包含了自然数集ω，且实际上推广了定理 5.4 中从ω开始反复进行取幂集运算很多次后得到的集合S。(见第十节，特别是定理 10.5。)此外，在数学实践中，不仅数论而且所有可用集合论实现的数学都能在这个典型模型中完成。因而，基础公理可视为奥卡姆剃刀原理(Occam's razor)的一个便利的应用。

定理 5.10 中所描述的情形代表了把线性序进一步定义为良序的十分普遍的等价形式。

定理 5.12 *一个线性序$\langle A, <\rangle$是良序的，当且仅当它没有无穷递减的链，即不存在A中的ω-序列σ，对所有$i\in\omega$，满足$\sigma(i+1)<\sigma(i)$。*

证明 其中一个方向是显然的。如果σ是一个递减链，那么$\{\sigma(i)\mid i\in\omega\}$是$A$的一个非空子集且无最小元。(对于某个$i$，任何元素都是$\sigma(i)$且$\sigma(i+1)$是更小的元素。)对于另一个方向，假设存在一个没有最小元的A的非空子集C。必须使用选择公理。它提供了一个定义域为$\{B\mid B\subseteq C\wedge B\neq\varnothing\}$的函数$f$，满足对每个非空$B\subseteq C$，$f(B)\in B$。现在递归定义一个函数$g$，满足$g(0)=f(C)$且$g(n+1)=f(\{y\mid y\in C\wedge y<g(n)\})$。注意到，如果任一集合$\{y\mid y\in C\wedge y<g(n)\}$是空的，那么$g(n)$将是$C$的最小元，这与假设矛盾。因而$f$总

有定义，并且可以利用定理 4.9 保证这样一个函数 g 的存在性。显然由 g 的定义，对每个 i，$g(i+1)<g(i)$。所以它就是所要求的递减链。 □

习题

1. 补充定理 5.4 的证明细节。
2. 如果 $\langle A, <_A\rangle$ 和 $\langle B, <_A\rangle$ 是偏序，定义 $A\times B$ 上的字典序 $<_{A\times B}$ 为 $\langle a, b\rangle <_{A\times B} \langle c, d\rangle$，当且仅当 $a<_A c$ 或 $a=c$ 且 $b<_B d$。定义 A 和 B 的无交并 $A\sqcup B = A\times\{0\}\cup B\times\{1\}$ 上的序 $<_{A+B}$ 为 $\langle a, 0\rangle <_{A+B} \langle a', 0\rangle$，当且仅当 $a<_A a'$；$\langle b, 1\rangle <_{A+B} \langle b', 1\rangle$，当且仅当 $b<_B b'$；并且 $\langle a, 0\rangle <_{A+B} \langle b, 1\rangle$ 对每个 $a\in A$ 且 $b\in B$ 成立。
   (a) 证明：如果 $\langle A, <_A\rangle$ 和 $\langle B, <_A\rangle$ 都是线性序或者都是良序，那么 $\langle A\times B, <_{A\times B}\rangle$ 也是。(这个序称为序 B 和 A 的积(product)。如果它们都是线性序，那么这个序含有“A 那么多个 B 的拷贝”。)
   (b) 证明：如果 $\langle A, <_A\rangle$ 和 $\langle B, <_A\rangle$ 都是线性序或者都是良序，那么 $\langle A\sqcup B, <_{A+B}\rangle$ 也是。(这个序称为序 A 和 B 的和(sum)。如果它们都是线性序，那么这个序含有一个 A，A 后面跟着一个 B。)
3. 令 $\mathcal{Z}$ 为第四节习题 9 中定义的整数。证明：存在一个关系 R，良序化 $\mathcal{Z}$。(当然，这个关系不可能是整数上通常的序。)
4. 令 $\mathcal{Q}$ 为第四节习题 10 中定义的有理数。证明：存在一个关系 R 良序化 $\mathcal{Q}$。(提示：利用下面的事实：每一个 $\mathcal{Q}$ 的元素 $[\langle a, b\rangle]$ 都有一个标准表示，它是等价类的一个元素，其中 a 和 b 互素，即它们在 $\mathcal{N}$ 中除 1 外没有公因子。)

   这些习题还与第一章第一节的习题 1 ~ 9 有关。

第六节　谓词逻辑中的策梅洛-弗兰克尔集合论

本节是为那些已经学过(带等号的)谓词逻辑，并且想要了解用谓词逻辑规范化的集合论语言写出的集合论公理的人准备的。在第一节到第五节中，以策梅洛 1908 年的风格和思想表述了公理 1 ~ 10。我们认为这样最能解释其内容和起源。本节中的形式化的表述和策梅洛的表述主要不同在于子集构造公理和替换公理，其中所允许的性质限制为能用以 $\in$ 和 $=$ 作为原始谓词的谓词逻辑语言表达的性质。在这样的格式中，公理如下：

公理 1(外延性公理)　$(\forall x)(\forall y)[(\forall z)(z\in x\leftrightarrow z\in y)\leftrightarrow(x=y)]$

公理 2(空集公理)　$(\exists x)(\forall y)[\neg(y\in x)]$

公理 3(无序对公理)　$(\forall x)(\forall y)(\exists z)(\forall w)[(w\in z)\leftrightarrow((w=x)\vee(x=y))]$

公理 4(并集公理)　$(\forall x)(\exists y)(\forall z)[(z\in y)\leftrightarrow(\exists u)((z\in u)\wedge(u\in x))]$

下面的公理实际上是一个公理模式，所谓公理模式是指某种公式形式，使得具有该形式的任何公式都是一个公理。

公理 5(子集构造或概括公理)　对每个公式 $\varphi(v, w_1, \cdots, w_n)$，其中 $v, w_1, \cdots, w_n$ 是它所有的自由变元，都有公理：

$$(\forall w_1)\cdots(\forall w_n)(\forall x)(\exists y)(\forall z)[(z\in y)\leftrightarrow((z\in x)\wedge\varphi(z,w_1,\cdots,w_n))]$$

公理 6(幂集公理)　$(\forall x)(\exists y)(\forall z)[((\forall w)((w\in z)\rightarrow(w\in x)))\leftrightarrow(z\in y)]$

公理 7(无穷公理)　$(\exists x)[(\exists z)((z\in x)\wedge((\forall w)(\neg(w\in z))))\wedge((\forall y)((y\in x)\rightarrow(\exists t)[t\in x\wedge(\forall u)[(u\in t)\leftrightarrow(u\in y\vee u=y)]]))]$

下面的公理实际上也是一个模式。

公理 8(替换公理)　对每个公式 $\varphi(x, y, w_1, \cdots, w_n)$，其中 $x, y, w_1, \cdots, w_n$ 是它所有的自由变元，都有公理：

$$(\forall w_1)\cdots(\forall w_n)$$
$$[(\forall x)(\forall y)(\forall z)(\varphi(x,y,w_1,\cdots,w_n)\wedge\varphi(x,z,w_1,\cdots,w_n)\rightarrow y=z)]\rightarrow$$
$$(\forall z)(\exists w)(\forall y)[y\in w\leftrightarrow(\exists x)((x\in z)\wedge\varphi(x,y,w_1,\cdots,w_n))]$$

为清楚地表达下一个公理介绍一些缩写。用 $pr(z)$，$pr_0(z,\ u)$，$pr_1(z,\ v)$ 和 $func(f)$ 分别来简记公式：z 是一个有序对，z 是第一投影为 u 的有序对，z 是第二投影为 v 的有序对和 f 是一个函数。把用谓词逻辑公式明确地表达这些性质留作习题 7。利用这些符号，写出选择公理如下：

公理 9(选择公理) $(\forall u)[(\exists y)(y\in u)\wedge(\forall y_1)(y_1\in u\rightarrow(\exists y_2)(y_2\in y_1))\rightarrow(\exists f)(func(f)\wedge(\forall z)(z\in f\rightarrow(\exists y)(y\in u\wedge pr_0(z,\ y))\wedge(\forall y)(y\in u\rightarrow(\exists z)(z\in f\wedge pr_0(z,\ y))\wedge(\forall z)(\forall y)(\forall w)(z\in f\wedge pr_0(z,\ y)\wedge pr_1(z,\ w)\rightarrow w\in y))]$

公理 10(基础公理) $(\forall x)[(\exists y)(y\in x)\rightarrow(\exists y)((y\in x)\wedge(\forall z)\neg((z\in x)\wedge(z\in y)))]$

谓词逻辑中基于这些公理的理论称为带选择公理的策梅洛-弗兰克尔集合论(ZFC)。如果去掉选择公理，相应的理论称为策梅洛-弗兰克尔集合论(ZF)。

习题

对习题 1 ~ 6 中列出的每个集合，给出一个其存在性的证明，其中所有子集构造公理或替换公理的应用都是对于带 $\in$ 和 $=$ 的谓词逻辑公式所定义的性质而言的：

1. 第四节习题 1 的集合 R。
2. 定理 5.4 的集合 S。
3. 从 A 到 B 的函数集合 B^A。
4. 对任何 $n\in\omega$，所有 ω 上长度为 n 的序列的集合。
5. ω 上的所有线性序的集合。
6. ω 上的所有良序的集合。
7. 用谓词逻辑公式定义下面每一个关系 $pr(z)$，$pr_0(z,\ u)$，$pr_1(z,\ v)$ 和 $func(f)$，它们分别表示 z 是一个有序对，z 是第一投影为 u 的有序对，z 是第二投影为 v 的有序对和 f 是一个函数。

第七节　基数：有穷和可数

在下一节着手系统研究序数和基数算术之前，先介绍关于有穷和可数集合大小的基本概念。这些是与本书以及数学和计算机科学所需基础课程相关的话题。两个集合等势，即有相同大小或基数：它们之间存在一个一一对应，这个基本概念是现在众所周知的甚至被认为是理所当然的。

定义 7.1 *两个集合 A 和 B 称为***等势的**(equinumerous)*(有相同大小或基数)，记为 $|A|\equiv|B|$，如果存在一个双射 f: $A\rightarrow B$。*

易见，这个概念具有许多我们所期望的性质。

命题 7.2 *等势关系是一个等价关系。*

证明 需要验证这个关系是自反的、对称的和传递的。从任何集合 A 到其自身的恒等映射证明了自反性。对称性可以由下面的事实(第三节习题 8(c，e))得到，即如果 f: $A\rightarrow B$ 是一个双射，那么它的逆 f^{-1}：$B\rightarrow A$ 也是一个双射。类似地，传递性可由下面的事实(第三节习题 8(d))得到，即两个双射 f: $A\rightarrow B$ 和 g：$B\rightarrow C$ 的复合是一个 A 到 C 的双射。 □

与两个集合具有相同大小的概念相伴的一个概念是比较不同集合的大小。称 A 的元素不比 B 多，$|A|\leqslant|B|$，如果 A 和 B 的部分(但或许不是全部)之间存在一个对应，也就是

一个单射(一一映射)f: $A \to B$。相反地，可能想定义逆关系 $|B| \geq |A|$ 为：存在一个满射(映满映射)g: $B \to A$。这些关系有一些简单的事实，把它们留作习题。(例如，它们都是自反的和传递的(习题1)并且 $|A| \leq |B|$，当且仅当 $|B| \geq |A|$ (习题2)。)尽管如此，像其他事情，比如很自然的问题，是否 $|A| \leq |B|$ 且 $|B| \leq |A|$ 蕴涵 $|A| \equiv |B|$，还有是否可以从"$\equiv$"定义的等价类中选出标准的代表元且不是平凡的，我们把这些问题留到第十一节关于序数和基数的一般讨论中。在这一节里，只讨论有穷和可数的集合(现在必须给出这些集合的定义)，所有事情对它们来说都很简单。我们"知道"什么集合应该是有穷的——那些和一个自然数 n 有相同大小的集合。

定义 7.3 一个集合 A 是**有穷的**(finite)，当且仅当对某个 $n \in \omega$，存在一个双射 f: $n \to A$。否则，A 就是**无穷的**(infinite)。类似地，一个序列 σ 是**有穷的**，如果它的定义域是某个 $n \in \omega$；否则，它是**无穷的**。

因而自然数是有穷集合大小的典型代表。毕竟，$n = \{0, 1, 2, \cdots, n-1\}$ 有 n 个元素。当然，由这些 $n \in \omega$ 所表示的大小是互不相同的，因而确切地说这些是有穷基数。事实上，对 n, $m \in \omega$，定义在 $|n|$ 和 $|m|$ 上的序 $\leq$ 和 $\geq$ 跟自然数通常的序是一致的。

定理 7.4 对 n, $m \in \omega$，存在一个单射 f: $n \to m$，当且仅当 $n \leq m$；存在一个满射 g: $m \to n$，当且仅当 $m \geq n$；因此存在一个双射 h: $n \to m$，当且仅当 $n = m$。

证明 最后一个断言可由前两个及通常的序 $<$ ($\in$)是 ω 的线性序(定理4.7)得到。由于 $n \leq m$ 蕴涵 n 是 m 的子集，所以前两个断言的"充分性"方向可立刻得到。而且，如果 $n > m$ 且存在一个单射 f: $n \to m$，那么任意延拓 f^{-1} 到 m 上没有定义部分，可以得到从 m 到 n 的满射。因此，只需要证明对每个 $m \in \omega$，对任何 $n > m$，不存在满射 g: $m \to n$。如果存在这样的满射，那么显然 $n = m+1$ 时就存在一个满射。归纳证明不存在满射 g: $m \to (m+1)$。对 $m = 0$ 是平凡的，因为 0 是空集，所以不能把 $1 = \{\varnothing\}$ 映满。假设对 m 不存在这样的映射，并且假设存在一个映射 g: $(m+1) \to (m+2)$。所以 $g(m) = i < m+2$。如果 $i = m+1$，那么 $h \upharpoonright m$ 是一个与归纳假设矛盾的满射。否则，选取 $j \neq i$，使得 $g(j) = m+1$，并且若 $k < m$ 且 $k \neq j$，定义 h: $m \to (m+1)$ 为 $h(j) = i$ 且 $h(k) = g(k)$。显然 h 是一个满射，这样就得到所需的矛盾。 □

接下来证明上面提到的关于有穷集合的大小关系的一些简单事实。

定理 7.5

(i)一个集合 A 是有穷的，当且仅当存在一个 $n \in \omega$ 和一个满射 g: $n \to A$，当且仅当存在一个 $m \in \omega$ 和一个单射 f: $A \to m$。

(ii)如果 A 和 B 是有穷的，那么存在一个单射 f: $A \to B$($|A| \leq |B|$)，当且仅当存在一个满射 g: $B \to A$($|B| \geq |A|$)。

(iii)如果 A 和 B 是有穷的，$|A| \leq |B|$ 且 $|B| \leq |A|$，那么 $|A| \equiv |B|$。

证明 对(i)，只需要证明：存在一个满射 g: $n \to A$ 或一个单射 f: $A \to n$，从而 A 是有穷的。由于可以很容易地从一个单射 f: $A \to n$ 定义一个满射 g: $n \to A$(如定理7.4的证明一样)，只需要证明，如果存在这样一个满射，那么 A 就是有穷的。考虑集合 $B = \{i < n \mid$ 对任何 $j < i$, $g(i) \neq g(j)\}$。显然 f 在 B 上的限制定义了一个双射。另一方面，可以很容易通过令 $h(j)$ 等于 B(在递增序下)的第 j 个元素，来定义一个从某个 $m \leq n$ 到 B 的双射 h。

(ii)和(iii)从定理7.4很容易得到，留作习题4。 □

下面列出有穷集合算术的一些简单而普通的事实，其余的在习题中给出。

定理 7.6 如果所有给出的集合都视为有穷的，那么下面的结论成立：

(i)$A\cup B$ 是有穷的(有穷集合的并是有穷的)。

(ii)$A\times B$ 是有穷的(有穷集合的积是有穷的)。

(iii)如果 σ 是一个有穷集合的有穷序列，即对每个 $i\in\mathrm{dom}\sigma$，$\sigma(i)$ 是有穷的，那么 $\bigcup\{\sigma(i)\mid i\in\mathrm{dom}\sigma\}$ 是有穷的(有限个有穷集合的并是有穷的)。

(iv)如果 σ 是一个有穷集合的有穷序列(长度为 n)，那么乘积 $\prod\{\sigma(i)\mid i\in\mathrm{dom}\sigma\}=\sigma(0)\times\sigma(1)\times\cdots\times\sigma(n-1)$ 是有穷的(有限个有穷集合的积是有穷的)。

证明 先考虑(i)和(ii)。假设 $f: n\to A$ 和 $g: m\to B$ 是双射。定义映射 $h_1:(n+m)\to A\cup B$ 和 $h_2: n\cdot m\to A\times B$ 如下：

(a)若 $i<n$，则 $h_1(i)=f(i)$ 且对 $n\leqslant i<n+m$，有 $h_1(i)=g(i-n)$。

(b)$h_2(i)=\langle f(j), g(k)\rangle$，其中 $i<n\cdot m$，j 是满足 $x\cdot m<i$ 和 $mx+k=i$ 的最大的 x。

显然，h_1 是一个满射。给定任何 $j<n$ 和 $k<m$，$h_2(j\cdot m+k)=\langle f(j), g(k)\rangle$，因而，$h_2$ 也是一个满射。这就证明了(i)和(ii)。(iii)和(iv)可由对 σ 长度的归纳立即得到。 □

不难计算定理 7.6 中提到的集合的确切大小。(见习题 5 ~ 7。)习题 8 ~ 9 提供了关于指数的类似事实。

因而，对于有穷集合，我们关于比较相对大小和(算术)运算的直觉似乎被验证了。当然，用有穷集合来验证我们的直觉，这并不奇怪。一般说来，无穷集合的运算不太符合我们的直觉。我们从不大于 ω 的集合开始研究。

定义 7.7 一个集合 A 是**可数的**(countable)，如果存在一个满射 $f: \omega\to A$。如果 A 也是无穷的，那么称它为**可数无穷的**(countably infinite)。如果 A 不是可数的，那么称它为**不可数的**(uncountable)。

注意，我们已经把有穷集合包含在那些可数的集合里面了(毕竟，可以数出它们)。第一点要指出的是存在可数无穷的集合。

命题 7.8 ω 是一个无穷集合(因而由定义，ω 是可数无穷的)。

证明 如果存在一个满射 $f: n\to\omega$，那么就有 $f: n\to(n+1)$ 与定理 7.4 矛盾。 □

接下来一点是可数无穷集合只有唯一的基数。

命题 7.9 如果 A 是可数无穷的，那么 $|\omega|\equiv|A|$。

证明 根据假设，存在一个满射 $f: \omega\to A$，但是，(由定理 7.5(i))从任何 $n\in\omega$ 出发都不是满射。令 $B=\{n\in\omega\mid$ 不存在 $m<n$，满足 $f(m)=f(n)\}$。显然，限制到 B 上的 f 是一个映满 A 的双射。而且，如果 B 在 ω 中是有界的，比如说 n 是其上界，那么这个限制将给出一个从 n 到 A 的满射，与假设矛盾。因而 B 是无界的，所以只需要证明：给定任何无界的 $C\subseteq\omega$，存在一个双射 $g: \omega\to C$。只要递归地定义 g，满足 $g(0)$ 是 C 的最小元素，并且 $g(i+1)$ 是 C 中比 $g(i)$ 大的最小元素。(这样的元素存在，因为 C 在 ω 中是无界的。)因此，根据定理 4.9，所需的函数 g 是存在的。 □

现在对可数集合建立类似定理 7.6(i) ~ (ii)的结论。

定理 7.10 如果 A 和 B 是可数的，那么

(i)$A\cup B$ 是可数的(可数集合的并是可数的)。

(ii)$A\times B$ 是可数的(可数集合的积是可数的)。

证明 假设 $f: \omega\to A$ 和 $g: \omega\to B$ 是满射。定义 ω 上的函数 h_1 和 h_2 如下：

(a)$h_1(2n)=f(n)$且$h_1(2n+1)=g(n)$。

(b)若$n=2^x\cdot 3^y$，则$h_2(n)=\langle f(x),\ g(y)\rangle$；否则，$h_2(n)=\langle f(0),\ g(0)\rangle$。

立即可得这些函数是满射，因而说明了$A\cup B$和$A\times B$的可数性。 □

对于定理7.6(iii)～(iv)的类似结论，情况更复杂些。给定可数集合的一个ω-序列，如果同时拥有序列中每一个集合的计数，那么就能给出它们的并集的计数。

定理7.11 如果σ是集合的一个ω-序列且τ是序列σ中的集合计数的ω-序列，即对每个$i\in\omega$，映射$\tau(i)$：$\omega\to\sigma(i)$是一个满射，那么$\bigcup\{\sigma(i)\mid i\in\mathrm{dom}\sigma\}$是可数的(可数个可数集合的并是可数的)。

证明 设p_i以递增的次序列举了素数(2，3，5，7，…)。利用数论的基本事实：若$p_i^x=p_j^y$，则$i=j$且$x=y$。定义一个映射h：$\omega\to\bigcup\{\sigma(i)\mid i\in\mathrm{dom}\sigma\}$为$h(p_i^x)=\tau(i)(x)$且对于不是素数幂的$n$，定义$h(n)=\sigma_0(0)$。显然，$h$是一个满射，因此，$\bigcup\{\sigma(i)\mid i\in\mathrm{dom}\sigma\}$是可数的。 □

仅当避免使用选择公理时，给出计数$\tau(i)$和集合$\sigma(i)$才是一个问题。对于一个给定的可数集合的ω-序列σ，选择公理将提供一个函数g来对每个集合$\sigma(i)$选出一个计数(习题10)，这个g就是定理中所需要的序列τ。如果从集合论公理中去掉选择公理，那么为说明可数集合的可数并还是可数的，需要在定理中明确地假设τ。没有它定理也许不成立。

最后，考虑可数个可数集合乘积的大小，这类似于定理7.6(iv)中对有穷集合的处理。第一个问题就是无穷的乘积是什么意思。有穷积如$\prod\{A_i\mid i<n\}=A_0\times A_1\times A_2\times\cdots\times A_{n-1}$，是由$n$元组组成，并且在这个积和$n$-序列$\sigma(\sigma(i)\in A_i)$之间存在一个明显的自然对应。因此，对一个集合的$\omega$-序列$\sigma$的乘积，自然会考虑相应的$\omega$-序列的集合。

定义7.12 如果σ是一个ω-序列，那么积$\prod\{\sigma(i)\mid i\in\omega\}$是指对每个$i\in\omega$，满足$\tau(i)\in\sigma(i)$的所有$\omega$-序列$\tau$的集合。如果存在一个集合$A$，满足对每个$i\in\omega$，有$\sigma(i)=A$，那么迭代的积像通常一样写成一个指数$A^\omega$。(注意到，这和原来把$A^\omega$定义为以$\omega$为定义域，值域包含在$A$中的所有函数集是一致的)。

特别地，考虑序列σ的积，其中每个$\sigma(i)=2=\{0,\ 1\}$，因此这个积是2^ω。通过匹配函数h：$\omega\to 2$和集合$A=\{n\in\omega\mid f(n)=1\}$，这个集合自然地对应于$\mathcal{P}(\omega)$。这个对应显然是一个双射，所以$2^\omega$和$\mathcal{P}(\omega)$具有相同的大小。在本节的最后，给出康托的著名定理：这些集合是不可数的。证明的方法称为对角线方法。

定理7.13 2^ω是不可数的，$\mathcal{P}(\omega)$也是不可数的。

证明 第二个断言来自于第一个断言和上面提到的那个双射。为了导出矛盾，假设存在满射f：$\omega\to 2^\omega$，因此，2^ω是可数的。定义$h\in 2^\omega$为：若$f(n)(n)=1$，则$h(n)=0$且若$f(n)(n)=0$，则$h(n)=1$。(记住每个$f(n)$是一个从ω到2的映射。)由于f是一个满射，存在一个n，满足$f(n)=h$。然而，根据对h的定义，$h(n)\neq f(n)(n)$，得到矛盾。 □

若把这个证明想象成是先把函数$f(n)$：$\omega\to 2$列成如下一个无穷的方阵

$$\begin{array}{ccccccccc}
f(0)(0) & f(0)(1) & f(0)(2) & \cdots & \cdots & f(0)(n) & \cdots & \cdots \\
f(1)(0) & f(1)(1) & f(1)(2) & \cdots & \cdots & f(1)(n) & \cdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
f(n)(0) & f(n)(1) & f(n)(2) & \cdots & \cdots & f(n)(n) & \cdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{array}$$

则定义不在列表上的函数 h 是通过沿着方阵对角线改变每个 $f(n)(n)$ 的值以确定 $h(n)$ 而构造出来的。因此，这个证明被称为对角线方法(diagonal argument)。(在第三章第八节中用它来证明停机问题是不可判定的(定理 8.8)。)习题 11 中给出定理 7.13 的更一般版本，这在第十一节(定理 11.9 和定理 11.14)中也有讨论。

习题

1. 证明：集合 A 和 B 间的二元关系 $|A| \leqslant |B|$ 和 $|B| \geqslant |A|$ 是自反的和传递的。
2. 证明：对任何集合 A 和 B，$|A| \leqslant |B|$ 蕴涵着 $|B| \geqslant |A|$。
3. 证明：对任何集合 A 和 B，$|A| \geqslant |B|$ 蕴涵着 $|B| \leqslant |A|$。(这个习题需要选择公理，对每个 $b \in B$，从 $\{a \in A \mid f(a) = b\}$ 中选择一个元素，其中 f 是由假设 $|A| \geqslant |B|$ 保证的满射。)
4. 利用定理 7.4 证明：定理 7.5 的(ii)和(iii)(即不用习题 2～3 的一般性结果)。
5. 证明：如果对 $n, m, k \in \omega$，有 $|A| \equiv |n|$，$|B| \equiv |m|$ 且 $|A \cap B| \equiv |k|$，那么 $|A \cup B| \equiv |n+m-k|$ 且 $|A \times B| \equiv |n \cdot m|$。
6. 证明：如果 σ 是一个两两不相交集合的有穷序列(即对 $i \neq j$，有 $\sigma(i) \cap \sigma(j) = \varnothing$ 且 $|\sigma(i)| \equiv |n_i|$，那么 $|\cup\{\sigma(i) \mid i \in \mathrm{dom}\sigma\}| \equiv |\Sigma\{n_i \mid i \in \mathrm{dom}\sigma\}|$。
7. 证明：如果 σ 是一个有穷集合的 m－序列，满足 $|\sigma(i)| \equiv |n_i|$，那么 $|\prod\{\sigma(i) \mid i \in \mathrm{dom}\sigma\}| \equiv |n_0 \cdot n_1 \cdot \ldots \cdot n_{m-1}|$。
8. 证明：如果 A 和 B 是有穷的，那么 B^A 也是有穷的。
9. 证明：如果对 $n, m \in \omega$，有 $|A| \equiv |n|$ 且 $|B| \equiv |m|$，那么 $|B^A| \equiv |m * n|$，其中 $*$ 表示定理 4.10 中定义的自然数上常用的指数函数。
10. 利用选择公理证明：给定一个可数集合的 ω－序列 σ，存在一个 ω－序列 τ，满足对每个 $i \in \omega$，$\tau(i)$ 是一个从 ω 到 $\sigma(i)$ 的满射。
11. (不用选择公理)证明：对任何集合 A
    (a)存在从 2^A 到 $\mathcal{P}(A)$ 的双射。
    (b)存在 A 到 2^A 的单射和 2^A 到 A 的满射。
    (c)不存在满射 $g: A \to 2^A$ 和单射 $f: 2^A \to A$。
    因此可知，对每个集合 A，$|A| \leqslant |2^A|$ 但 $|2^A| \nleqslant |A|$(并且类似有 $|2^A| \geqslant |A|$ 但 $|A| \ngeqslant |2^A|$)。由(a)这些结果对 A 和 $\mathcal{P}(A)$ 同样正确。把这些事实说成是 A 的幂集 $\mathcal{P}(A)$ 的基数严格大于 A 的基数。这个一般性结果就是康托定理。

第八节　序　　数

在余下的几节中，考虑任意集合的序和大小的概念。我们从序数开始。康托认为，就像可以按顺序数完一个有穷(或可数的)集合并能用自然数穷举它一样，也可以按顺序数完一个任意集合并能用广义的自然数来穷举它。他称这些新的数为序数。他认为完全能够“数出”任何集合。他的良序原理的内容就是这是可能的。这是序数理论的部分动机。

普通的计数从自然数 ω 开始：0，1，2，…。康托从第一个无穷序数 ω 继续下去，并且继续数 $\omega+1$，$\omega+2$，…。在“所有这些数”之后他继续数 $\omega \cdot 2$，$\omega \cdot 2+1$，$\omega \cdot 2+2$，…。他称一个序数为一个*极限序数*(limit ordinal)，如果它不是 0 且没有直接前驱。他称一个序数为一个*后继序数*(successor ordinal)，如果它有直接前驱。因此，ω 和 $\omega \cdot 2$ 是前两个极限序数。以上列出的所有其他序数(除了 0)都是后继序数。

康托关于构造序数的原理是：如果已经构造了一个序数的集合，那么在此集合之外存在一个最小的序数。他称这个原理为“第一构造原理”。因此，每个序数都作为前面序数的集合之后的下一个序数被引进。每个新序数引进时，如果前面所有序数集合没有最大元，它作

为前面序数的极限序数，如果前面所有序数的集合有最大元，它就作为前一个序数的后继序数。

这并不是序数的定义，因为它是循环定义的。两边都有“序数”这一项。“前面的”这个项，即序数的次序，没有解释。要按照康托的解释构造它们，你就必须先知道什么是序数。这不是康托的不足，因为他认为序数的存在就像自然数的存在一样是直觉上显然的。他没有定义它们，因为对他来说，序数和自然数一样是直接的直觉能解释的东西。

von Neumann[1923, 2.3]则用一个序数的定义形式化了康托的意图，该定义规定每个序数都是所有较小序数的集合。(在第四节中所定义的自然数正是冯·诺伊曼序数的开始部分。)因此，他对于上面所列出的序数的想法是：

$$0=\varnothing$$
$$1=\{0\}$$
$$2=\{0,1\}$$
$$\vdots$$
$$\omega=\{0,1,2,\cdots\}$$
$$\omega+1=\{0,1,2,\cdots,\omega\}$$
$$\omega+2=\{0,1,2,\cdots,\omega,\omega+1\}$$
$$\vdots$$
$$\omega\cdot 2=\{0,1,2,\cdots,\omega,\omega+1,\cdots\}$$
$$\omega\cdot 2+1=\{0,1,2,\cdots,\omega,\omega+1,\cdots,\omega\cdot 2\}$$
$$\omega\cdot 2+2=\{0,1,2,\cdots,\omega,\omega+1,\cdots,\omega\cdot 2+1\}$$
$$\vdots$$

但是，这些仅仅是例子，不能作为序数的一般定义。要求每个序数是前面所有序数的集合也是循环的。所以，冯·诺伊曼寻求以非循环的方式定义序数，以便从它们形式化的定义，他能证明每个序数就是前面所有序数的集合。这样的话，他就“打破了”循环定义中的“循环”。

下面看看他是如何做到的。首先，如果每个序数都是前面序数的集合，那么对序数来说 $a<b$ 就和 $a\in b$ 一样了。所以，他预先知道了序数的序应该是怎么样的。他还发现，如果每个序数都是前面序数的集合，那么序数应该具有下面列出的性质：

1. 由于序数的序应当是传递的，如果 y 是一个序数且 a 是(小于)b 的前驱，b 是 y 的前驱，即 $a\in b\in y$，那么 a 应当是 y 的前驱序数，即 $a\in y$。

2. 由于序数应当是线性序的，给定任意两个不同的序数，一个应该在另一个前面。即一个应该是另一个的元素。

3. 如果 y 是一个序数并且 $A\subseteq y$，$A\neq\varnothing$，那么通过“构造”y 中每个序数为前面序数的集合，从 0 开始，我们应当会最终首次构造出 A 中的一个元素。因此，一个序数的任何非空子集都会有一个最小元。如果序数的序是 $\in$，这确切地说就是一个序数的任何非空子集 A 都会有一个元素 a 与 A 不相交(即 $a\cap A=\varnothing$)。

现在可以不管这些性质，给出冯·诺伊曼关于序数的非循环定义。

定义 8.1

1. **一个集合 A 是传递的**，如果 $z\in w$ 且 $w\in y$ 从而 $z\in y$。

2. 一个集合 y 是**连通的**(connected)，如果 $z\in y$，$w\in y$ 且 $z\neq w$ 从而 $z\in w$ 或 $w\in z$。

3. 一个集合 y 是**良基的**(well founded)，如果当 $A\subseteq y$ 且 $A\neq\varnothing$ 时，就存在 A 的一个 **∈-极小元**，即存在一个 $x\in A$，满足 $x\cap A=\varnothing$。

定义 8.2 一个集合 y 是一个**序数**，如果 y 是传递的、连通的和良基的。(显然，后两个条件也可以用 $\in$ 良序化 y 来替换。)

注意到，按这个定义，上文列出的那些特殊的例子确实是序数。

评注 8.3 每个集合都是良基的，这个命题就是基础公理(公理 10)。通过明确的要求(关于隶属关系的)良基性作为一个序数定义的一部分，我们可以不用这个公理来发展序数和基数的理论。这一观点与第十节有关，在那里我们把第四节习题 1 和第六节习题 1 中的层次 $R(n)$ 推广至所有序数，并且证明了(在没有基础公理的集合论中)所有这些集合的并是包括基础公理在内的集合论的一个“模型”。这就给出了关于包含基础公理的集合论与不包含它的集合论的一个相对相容性结论。

记号 序数的序是 $\in$。如果 α，β 是序数，通常记 $\alpha<\beta$ 而不是 $\alpha\in\beta$，并且当 $(\alpha\in\beta)$ 或 $(\alpha=\beta)$ 时，记 $\alpha\leqslant\beta$。

现在从序数的形式化定义导出每个序数确实是前面序数的集合。

定理 8.4 如果 α 是一个序数，那么 $\alpha=\{\beta\mid\beta$ 是一个序数 $\wedge\beta\in\alpha\}$。

证明 显然，只需要证明每个 $\beta\in\alpha$ 都是一个序数。首先注意到，由 α 的传递性，$\beta\subseteq\alpha$。根据定义，一个连通的(或良基的)集合的每个子集都是连通的(或良基的)，所以只需要证明 β 是传递的。于是，考虑任何 $x\in y\in\beta$。由 α 的传递性，$y\in\alpha$。类似地，$x\in y\in\alpha$ 蕴涵 $x\in\alpha$。由 α 的连通性，或者 $x\in\beta$ 或者 $x=\beta$ 或者 $\beta\in x$。证明后两种情况不可能发生，因此 β 是传递的：

如果 $x=\beta$，那么 $\beta\in y\in\beta$。因而集合 $A=\{\beta, y\}\subseteq\alpha$ 没有与 A 不相交的元素，这与 α 的良基性矛盾。另一方面，如果 $\beta\in x$，那么 $\beta\in x\in y\in\beta$，这种情况下 $A'=\{\beta, x, y\}\subseteq\alpha$，违反了 α 的良基性。 □

现在证明所有序数的族 ON 具有所有我们对每个序数所要求的性质。

定理 8.5

1. ON 是传递的：如果 α 是一个序数且 $x\in y\in\alpha$，那么 x 是一个序数。
2. ON 是连通的：如果 α 和 β 是不同的序数，那么 $\alpha\in\beta$ 或 $\beta\in\alpha$。
3. ON 是良基的：如果 A 是一个序数的非空集合，那么存在 $x\in A$，满足 $x\cap A=\varnothing$。

证明 第一个命题两次应用定理 8.4 就可以证明。

对于第二个，假设 $\alpha\neq\beta$。因而，$\beta\not\subseteq\alpha$ 或 $\alpha\not\subseteq\beta$。这两种情况是对称的，所以假设 $\beta\not\subseteq\alpha$。由于 $\beta\not\subseteq\alpha$，$\beta-\alpha\neq\varnothing$。由 β 的良基性，存在一个 $\in$-极小元 $\gamma\in\beta-\alpha$(由定理 8.4，γ 是一个序数)。由 β 的传递性，任何 $x\in\gamma$ 都是 β 的一个元素，所以由 γ 的极小性，它是 α 的一个元素。另一方面，如果存在 $x\in\alpha-\gamma$，令 δ 为 $\alpha-\gamma$ 的 $\in$-极小元。正如我们所见，任何 $y\in\gamma$ 都在 α 中，所以，由 α 是连通的，$y=\delta$ 或 $\delta\in y$ 或 $y\in\delta$。第一个可能性与假设 $\delta\notin\gamma$ 矛盾。第二个与 γ 的传递性和对 δ 的选择矛盾。从而，对每个 $y\in\gamma$，有 $y\in\delta$，因此 $\gamma\subseteq\delta$。另一方面，根据 δ 的极小性，$y\in\delta$ 从而 $y\in\gamma$。所以，$\gamma=\delta$，这与选择的 δ 在 α 中而 γ 不在 α 中矛盾。

最后，为说明 ON 是良基的，考虑序数的任何非空集合 A。令 α 为 A 的任何元素。如果

$\alpha \cap A = \varnothing$，那么 α 就是所要的 A 的 $\in$-极小元。否则，$A \cap \alpha$ 的一个 $\in$-极小元(其存在性由 α 是一个序数保证)也是 A 本身的一个元素。 □

如果所有序数的族是一个集合，那么根据定理8.5，它将是一个序数。实际上，它将是最大的序数，因为每个序数都是它的一个元素。这显然是违反序数的直观性的。在集合论公理化之前，这个事实被认为是一个悖论，它由布拉里(Burali)和福蒂(Forti)提出。像罗素悖论一样，现在我们视它为一个定理。

定理8.6(布拉里-福蒂)*所有序数的族不是一个集合。*

证明 如果存在所有序数的一个集合 ON，那么根据定理8.5，ON 将是一个序数，所以 $ON \in ON$。这与序数 ON 的良基性矛盾。 □

有两种方法可以重新改写定理8.5中给出的序数族的良基性，但首先需要有证明序数性质的一种标准方法：超穷归纳证明。

定理8.7(超穷归纳证明) *令 P 是序数的一个性质。或者没有序数具有性质 P，或者存在一个最小的序数具有性质 P。等价地，如果对每个序数 α，有*

$$\text{如果对每个}\ \beta < \alpha, P(\beta)\ \text{成立,那么}\ P(\alpha)\ \text{也成立,} \qquad (*)$$

那么对每个序数 γ，$P(\gamma)$ 成立。

证明 假设 $P(\beta)$，并且 β 不是具有性质 P 的序数中最小的。那么由定理8.5(iii)，$A = \{x \in \beta \mid P(x)\}$ 有一个 $\in$-极小元。根据定理8.5(ii)，这个元素是最小的。注意到，不具有性质 P 的最小的 α 与归纳假设(*)矛盾，等价形式可立即得到。 □

如果没有上面所说的动机，那么序数的这种构造是很不直观的。一旦认识到关键是打破和消除定义中的循环性，这样构造的必要性就变得很显然而且也是有理由的。现在，既然有了序数真正的定义，我们可以把 ω 上的算术运算推广到超穷的情况。在下一节中将作这样的推广。

评注8.8 假设发展集合论时有一条有穷公理："所有集合都是有穷的"，用它取代无穷公理。我们需要同样的序数构造。但是能被证明存在的序数将仅仅是自然数 $\{0, 1, 2, \cdots\}$。在这样的集合论中，布拉里-福蒂定理说明不存在归纳集合，即不存在所有自然数的集合 ω。

评注8.9 关于序数理论，伯特兰·罗素更早地给序数理论提供了一个不同的基础。在他的理论中，序数不是集合。这和康托不同，康托的序数都是集合，正如上文冯·诺伊曼的理论一样。罗素引入序数作为良序集的"序型"。给定两个良序集 $\langle A, <_A\rangle$ 和 $\langle B, <_B\rangle$，罗素像康托一样称 $\langle A, <_A\rangle$ 相似于 $\langle B, <_B\rangle$，记为 $\langle A, <_A\rangle \sim \langle B, <_B\rangle$，如果存在一个1-1的满射 $f: A \to B$，满足对所有 $a_1, a_2 \in A$，$a_1 \leqslant_A a_2$，当且仅当 $f(a_1) \leqslant f(a_2)$。(这样的 f 称为两个良序集之间的序同构(order isomorphism)。)易见，$\sim$ 是自反的、对称的和传递的(习题1)。给定一个良序集 $\langle A, <_A\rangle$，罗素让 $\langle A, <_A\rangle$ 的序数是所有相似于 $\langle A, <_A\rangle$ 的良序 $\langle B, <_B\rangle$ 的集合。但是承认这是一个集合，会导出像布拉里-福蒂那样的悖论(习题5)。罗素发展他的"类型论"以绕开这样的麻烦。由于每个良序都等价于一个唯一的序数(习题4)，我们可以视冯·诺伊曼序数为罗素序数的典型表示。关于冯·诺伊曼序数算术和序型运算间的其他联系参见第九节习题8。

习题

1. 证明：在评注8.9中定义的良序间的相似性概念是一个等价关系。

2. 证明：如果 x 和 y 是序数且 $\langle x, \in\rangle \sim \langle y, \in\rangle$，那么 $x = y$。
3. 证明：如果 A 是一个序数的传递集合，那么 A 本身是一个序数。
4. 证明：如果 $\langle A, <_A\rangle$ 是一个良序，那么存在唯一的序数 $\langle\alpha, \in\rangle$，满足 $\langle A, <_A\rangle \sim \langle\alpha, \in\rangle$。（提示：令 $B = \{a \in A \mid$ 存在序数 x，满足 $\langle x, \in\rangle \sim \langle\{b \in A \mid b <_A a\}, <_A\rangle\}$，并且定义一个从 B 到 ON 的函数，它把 $a \in B$ 映到同构于 $\langle\{b \in A \mid b <_A a\}, <_A\rangle$ 的序数（根据习题 1 和 2，它是唯一的）。然后利用替换公理和 3。）
5. 证明：给定任何非空良序 $\langle B, <_B\rangle$，不存在由所有同构于它的良序 $\langle A, <_A\rangle$ 组成的集合。

第九节　序数算术和超穷归纳

现在有了自然数的超穷推广，我们可以通过发展序数算术理论扩展相关内容。和处理自然数时一样，我们的出发点是后继函数。每个序数都有一个直接后继，并且它由我们对冯·诺伊曼自然数使用过的同样的函数给出。

命题 9.1　如果 α 是序数，那么 $s(\alpha) = \alpha \cup \{\alpha\}$ 是序数，并且是 α 在序数上的 $\in$ 序下的直接后继。

证明　$s(\alpha)$ 成为序数所要满足的三条性质都可以由 α 的同样性质得到（习题 1）。由于 $\alpha \in s(\alpha)$，α 是 $s(\alpha)$ 的一个前驱。为了说明它是直接前驱，考虑任何 $\beta \in s(\alpha)$，$\beta = \alpha$ 或 $\beta \in \alpha$。正是我们想要的。□

因此，每个序数都有一个直接后继，但不同于自然数，存在不为 0 的序数没有直接前驱。第一个这样的序数是（冯·诺伊曼）自然数的集合 ω。命名这些序数是很有益的。

定义 9.2　一个序数 α 是一个**后继序数**，如果对某个序数 β，它具有 $s(\beta)$ 的形式，或等价地（由习题 2），如果在序数的序下它有直接前驱。一个序数 λ，它既不是 0 也不是后继序数，就称为**极限序数**。

极限序数是其下面的序数的并，并且也是它们下面序数集合的最小上界。并算子和序数的有穷集合上“max”函数是一致的，一般地，它和序数集合上最小上界算子是一致的。

命题 9.3　如果 α 和 β 是序数，λ 是极限序数且 x 是一个序数的集合，那么

1. $\alpha \cup \beta = \max\{\alpha, \beta\}$（即无论是 α 或 β，至少不小于另外一个）。
2. $\lambda = \bigcup\{\gamma \mid \gamma \in \lambda\}$。
3. $\bigcup x$ 是 x 的最小上界，即最小的序数 δ，对每个 $\gamma \in x$，满足 $\gamma \leqslant \delta$。

证明　留作习题 3。□

要进一步发展序数算术，关键是把递归定义方法（定理 4.9）推广到超穷序数上。事实上，这正是集合论的主要内容。

记号　$F \restriction \alpha$，F 在 α 上的限制，是函数 $\{\langle x, F(x)\rangle \mid x \in \alpha\}$。

定理 9.4（超穷递归或超穷归纳定义）：　如果 G 是一个定义在所有集合 X 上的函数性质，那么存在唯一的函数性质 F，它定义在所有序数 α 上，且对所有 α，满足

$$F(\alpha) = G(F \restriction \alpha)$$

证明　首先定义一个（F 的）逼近（approximation）为任何函数 f：它的定义域为一个序数 α，对所有 $\beta \in \alpha$，满足

$$f(\beta) = G(f \restriction \beta)$$

断言：如果 f_1 和 f_2（定义域分别为 α_1 和 α_2）都是逼近，那么 $f_1 \subseteq f_2$ 或 $f_2 \subseteq f_1$。为说明这一点，注意到，$\alpha_1 \leqslant \alpha_2$ 或 $\alpha_2 \leqslant \alpha_1$。假设 $\alpha_1 \leqslant \alpha_2$。证明 $f_1 \subseteq f_2$。否则，（根据定理 8.7）存在最小的

$\alpha\in\alpha_1$，满足$f_1(\alpha)\neq f_2(\alpha)$。那么对$\beta\in\alpha$，$f_1(\beta)=f_2(\beta)$。所以$f_1\upharpoonright\alpha=f_2\upharpoonright\alpha$且$G(f_1\upharpoonright\alpha)=G(f_2\upharpoonright\alpha)$。由于$f_1$和$f_2$都是逼近，我们断定$f_1(\alpha)=f_2(\alpha)$。由于这是一个矛盾，所以不存在这样的$\alpha$，从而$f_1\subseteq f_2$。类似地，$\alpha_2\leqslant\alpha_1$蕴涵了$f_2\subseteq f_1$。

现在可以定义F为$F(\alpha)=y$，如果存在一个逼近f，满足$f(\alpha)=y$。根据断言，F是一个函数性质。因为$F(\alpha)=y$蕴涵了对某个逼近f，有$f(\alpha)=y$，所以，当$F(\alpha)$有定义时，$F(\alpha)=G(F\upharpoonright\alpha)$。剩下的只需要证明$F(\alpha)$对每个序数$\alpha$都有定义。否则，存在最小的序数$\alpha$，满足$F(\alpha)$没有定义。但由替换公理，$F\upharpoonright\alpha$是一个集合，所以它实际上是一个逼近。如果$y=G(F\upharpoonright\alpha)$，那么$(F\upharpoonright\alpha)\cup\{\langle\alpha,y\rangle\}$也是一个逼近。它的定义域是$s(\alpha)=\alpha\cup\{\alpha\}$，所以$F$在$\alpha$处也有定义，得到矛盾。

最后，为说明F是唯一的，假设H也满足定理条件。如果存在某个序数α，满足$F(\alpha)\neq H(\alpha)$，那么就存在最小的一个。然而，对这个α，对每个$\beta<\alpha$，$F(\beta)=H(\beta)$。因此，$F\upharpoonright\alpha=H\upharpoonright\alpha$，从而由定理中定义的性质，有$F(\alpha)=G(F\upharpoonright\alpha)=G(H\upharpoonright\alpha)=H(\alpha)$，得到矛盾。 □

超穷递归的另一个简单实用的刻画是描述了在0、后继及极限序数等处该做什么。

推论 9.5 如果G和H是定义在所有集合上的函数性质且z是一个给定的集合，那么存在定义在所有序数上的唯一的函数性质F，满足

1. $F(0)=z$
2. $F(s(\alpha))=G(F(\alpha))$，对每个后继序数$s(\alpha)$
3. $F(\lambda)=H(F\upharpoonright\lambda)$，对每个极限序数$\lambda$。

而且，如果G和H是定义在所有序数上且值域包含在序数中的函数性质，z是一个序数，那么存在唯一的上述函数性质F，它的值也是序数。

证明 为证明第一个结论，只要定义一个函数性质J为：$J(0)=z$；$J(f)=G(f(\alpha))$，如果f是一个函数，它的定义域是一个后继序数，即为$s(\alpha)$，α为某个序数；$J(f)=H(f)$，如果f是一个函数，它的定义域是一个极限序数；否则$J(x)=0$。现在第一个结论可由定理9.4立即得到。对于第二个结论，只要拓展G和H，在不是序数的集合上定义为0，并应用第一个结论。当然，所有F的值或者为z或者为G和H的一个值，从而都是序数。 □

用这个推论提供的超穷递归形式来定义序数算术的函数。（记住，根据命题9.3(iii)，一个序数的集合的并是它的最小上界。）

定理 9.6 存在(唯一的)序数上的二元函数性质$+$，$\cdot$和$*$，满足下面的递归条件：

1. 序数加法

(a)$\alpha+0=\alpha$

(b)$\alpha+\beta=s(\alpha+\gamma)$，对每个后继序数$\beta=s(\gamma)$

(c)$\alpha+\lambda=\bigcup\{\alpha+\beta\mid\beta<\lambda\}$，对每个极限序数$\lambda$。

2. 序数乘法

(a)$\alpha\cdot 0=0$

(b)$\alpha\cdot\beta=\alpha\cdot\gamma+\alpha$，对每个后继序数$\beta=s(\gamma)$

(c)$\alpha\cdot\lambda=\bigcup\{\alpha\cdot\beta\mid\beta<\lambda\}$，对每个极限序数$\lambda$。

3. 序数指数

(a)$\alpha*0=1$

(b)$\alpha * \beta = (\alpha * \gamma) \cdot \alpha$，对每个后继序数 $\beta = s(\gamma)$

(c)$\alpha * \lambda = \bigcup\{\alpha * \beta \mid \beta < \lambda\}$，对每个极限序数 λ。

证明 每个函数的存在性和唯一性都由推论 9.5 立即得到。 □

注意到 $\alpha + 1 = s(\alpha)$ 由加法的定义立即可得(习题 4)。序数和序数算术更复杂的性质一般由超穷归纳证明(定理 8.7)。对应于推论 9.5 中超穷递归定义使用的把序数划分为 0、后继和极限序数的情况，存在一种超穷归纳的形式：首先(不使用任何假设)证明某个给定性质对于 0 成立，然后(假设该性质对于所有比较小的序数都成立)证明该性质对于较大的后继序数和极限序数成立，由此可得，该性质对于所有的序数成立。我们给出两个例子并把其他的留作习题。

命题 9.7 如果 $\beta \leqslant \gamma$，那么对所有序数 α，β 和 γ，有 $\alpha + \beta \leqslant \alpha + \gamma$。

证明 通过对序数 γ 归纳证明此命题。如果 $\gamma = \beta$，那么显然 $\alpha + \beta = \alpha + \gamma$。特别地，如果 $\gamma = 0$，那么 $\beta = 0$ 且 $\alpha + \beta = \alpha + \gamma$。现在对 γ 归纳证明：对所有 $\beta < \gamma$，有 $\alpha + \beta \leqslant \alpha + \gamma$。如果 γ 是一个后继序数 $s(\delta)$，那么 $\beta = \delta$ 或 $\beta < \delta$。第一种情况下，显然 $\alpha + \beta = \alpha + \delta < s(\alpha + \delta) = \alpha + \gamma$。第二种情况下，由归纳，$\alpha + \beta < \alpha + \delta$ 同时 $\alpha + \delta < s(\alpha + \delta) = \alpha + \gamma$。最后，如果 γ 是一个极限序数，那么 $\alpha + \gamma = \bigcup\{\alpha + \delta \mid \delta < \gamma\}$。特别地，由于 $\beta < \gamma$，$\alpha + \beta$ 是 $\{\alpha + \delta \mid \delta < \gamma\}$ 的一个元素，所以小于等于它的最小上界(由命题 9.3(iii)，就是它的并)。 □

命题 9.8(加法的结合律) 对所有序数 α，β，γ：$((\alpha + \beta) + \gamma) = (\alpha + (\beta + \gamma))$。

证明 对 γ 归纳。对 $\gamma = 0$，有 $((\alpha + \beta) + \gamma) = \alpha + \beta$ 且根据定义，$\beta + \gamma = \beta$，所以 $((\alpha + \beta) + \gamma) = \alpha + \beta = (\alpha + (\beta + \gamma))$，满足要求。然后假设 γ 是一个后继序数 $s(\delta)$。这种情况下，根据定义，$((\alpha + \beta) + \gamma) = s((\alpha + \beta) + \delta)$ 且 $\alpha + (\beta + \gamma) = \alpha + s(\beta + \delta) = s(\alpha + (\beta + \delta))$。根据归纳假设 $(\alpha + \beta) + \delta = \alpha + (\beta + \delta)$，有 $((\alpha + \beta) + \gamma) = (\alpha + (\beta + \gamma))$，即为所需。最后，如果 γ 是一个极限序数，那么根据定义，$(\alpha + \beta) + \gamma = \bigcup\{(\alpha + \beta) + \delta \mid \delta < \gamma\}$，同时根据归纳，对 $\delta < \gamma$，有 $(\alpha + \beta) + \delta = \alpha + (\beta + \delta)$，从而，由归纳有 $(\alpha + \beta) + \gamma = \bigcup\{\alpha + (\beta + \delta) \mid \delta < \gamma\}$。另一方面，根据定义，$\alpha + (\beta + \gamma) = \alpha + \bigcup\{\beta + \delta \mid \delta < \gamma\}$。现在，由于 γ 是一个极限序数，$\lambda = \bigcup\{\beta + \delta \mid \delta < \gamma\}$ 也是(习题 5)。因此，$\alpha + (\beta + \gamma) = \bigcup\{\alpha + v \mid v < \lambda\}$。因为 λ 是 $\{\beta + \delta \mid \delta < \gamma\}$ 的最小上界(命题 9.3)，所以 $\bigcup\{\alpha + v \mid v < \lambda\} = \bigcup\{\alpha + (\beta + \delta) \mid \delta < \gamma\}$。 □

虽然一些我们熟悉的自然数的算术定律对所有序数也成立(例如命题 9.7 和命题 9.8 中那些)，但有很多并非如此。例如，加法不是交换的，因为 $\omega + 1 \neq 1 + \omega = \omega$；乘法也是一样，因为 $\omega \cdot 2 \neq 2 \cdot \omega$。这些事实的证明和其他例子留作习题。关于序数加法和乘法的另一个重要事实(习题 9)是它们酷似第五节习题 2 中抽象定义的良序的加法和乘法。

习题

1. 证明：若 α 是一个序数，则 $s(\alpha)$ 也是。
2. 证明：一个序数 α 在序数的序下有直接前驱，当且仅当存在序数 β，满足 $s(\beta) = \alpha$。
3. 证明命题 9.3。
4. 证明：对每个序数 α，有 $\alpha + 1 = s(\alpha)$。
5. 证明：若 γ 是一个极限序数，则 $\bigcup\{\beta + \delta \mid \delta < \gamma\}$ 也是。
6. 证明下面关于序数乘法的事实：

   (a)$\alpha \cdot 0 = 0$。

   (b)$\alpha \cdot 1 = \alpha$。

   (c)若 $\beta \leqslant \gamma$，则 $\alpha \cdot \beta \leqslant \alpha \cdot \gamma$。

(d)(结合律)$\alpha\cdot(\beta\cdot\gamma)=(\alpha\cdot\beta)\cdot\gamma$。

7. 证明：对每个极限序数 λ 和每个 $n\in\omega$，$n\neq 0$，有 $n+\lambda=\lambda=n\cdot\lambda$。

8. 证明：$\alpha*(\beta*\gamma)=(\alpha*\beta)*\gamma$。

9. 我们说两个线性序 $\langle L, <_L\rangle$ 和 $\langle M, <_M\rangle$ 是同构的，如果存在一个一一的函数 f 从 L 映满 M，满足对每个 $a, b\in L$，有 $f(a)<_M f(b)$，当且仅当 $a<_L b$。

(a)证明：良序 $\langle\alpha+\beta, \in\rangle$ 和 $\langle\alpha\sqcup\beta, <_{\alpha+\beta}\rangle$(第五节习题 2 中定义)是同构的。

(b)证明：良序 $\langle\alpha\cdot\beta, \in\rangle$ 和 $\langle\alpha\times\beta, <_{\alpha\times\beta}\rangle$(第五节习题 2 中定义)是同构的。

10. 如果 $\langle T, <_T\rangle$ 是一棵树且 α 是一个序数，那么 T 的 α 层(level)由所有那些前驱的序型为 α 的 $x\in T$ 组成，即 $\langle\{y\mid y<_T x\}, <_T\rangle$ 同构于 $\langle\alpha, \in\rangle$。假设 T_1 和 T_2 都是树。证明：存在一棵树 $T_1\times T_2$，它的第 α 层是 T_1 和 T_2 的第 α 层的笛卡儿积，并且它的序 $<_{T_1\times T_2}$ 为 $\langle x, y\rangle<_{T_1\times T_2}\langle x', y'\rangle$，当且仅当 $x<_{T_1}y$ 且 $x'<_{T_2}y'$。

第十节 超穷递归、选择和有秩全域

本节介绍序数和超穷递归的两个应用。第一个是证明第五节中介绍的各种选择公理版本的等价性。第二个是通过幂集迭代把第四节习题 1 和第六节习题 1 中归纳定义的有限秩 $R(n)$ 推广至超穷序数并证明这样构造的集合就是整个全域。

我们从一个关于良序的引理开始。

引理 10.1 如果 $\langle A, R\rangle$ 是一个良序集且 h: $A\to B$ 是一个双射，那么 B 可以被良序化。

证明 令 S 为

$$\{\langle h(a_1),h(a_2)\rangle\in B\times B\mid\langle a_1,a_2\rangle\in R\}$$

那么对所有 a_1，$a_2\in A$，a_1Ra_2 当且仅当 $h(a_1)Sh(a_2)$。可直接证明 $\langle B, S\rangle$ 是良序的(习题 8)。 □

每个序数在 $\in$ 下都是良序的。所以，为证明每个集合都可以被良序化，根据引理 10.1，只需要证明对任何集合 B，存在一个序数 α 和一个 1-1 的映满的函数 h: $\alpha\to B$。现在重新叙述并给出定理 5.9 的完整证明。

定理 10.2 下面的断言是等价的：

(i)(选择公理)对每个非空集合的集合 A，存在一个定义域为 A 的函数 f，满足对每个 $x\in A, f(x)\in x$。

(ii)(良序原理)对每个集合 A，存在一个关系 $R\subseteq A\times A$ 良序化 A。

(iii)(佐恩引理)令 $\langle A, <\rangle$ 为任何非空偏序。如果每条链 $B\subseteq A$ 都有一个上界，那么 A 含有极大元。

证明 (i)$\Rightarrow$(ii)：假定(i)(选择公理)，希望构造一个给定集合 B 的良序。根据引理 10.1，只需找到一个序数 α 和一个函数 g: $\alpha\to B$，它是 1-1 和映满的。

令 $A=P(B)-\{\varnothing\}$。那么，根据选择公理，存在一个函数 f: $A\to\cup A$，满足对所有非空 $C\subseteq B$，$f(C)\in C$。超穷归纳定义所需函数 g，使得 $g(0)=f(B)$ 且当 $B-\{g(\beta)\mid\beta\in\alpha\}\neq\varnothing$ 时，$g(\alpha)=f(B-\{g(\beta)\mid\beta\in\alpha\})$，否则 $g(\alpha)=z$(其中 z 是某个不在 B 中的集合)。显然，(根据超穷归纳)对任何 $x\in B$，最多存在一个 α，满足 $g(\alpha)=x$。如果不存在序数 β，满足 $g(\beta)=z$，那么 g^{-1} 将是一个从 B 映满 ON 的 1-1 映射，由替换公理，这和定理 8.6 矛盾。因此，可以令 α 是满足 $g(\alpha)=z$ 的最小序数。注意到 $g\upharpoonright\alpha$ 是 1-1 的。现在断言 rg g 就是所需的整个 B。否则，由于 $B-\{g(\beta)\mid\beta\in\alpha\}\neq\varnothing$，那么 g 的定义说明 $g(\alpha)=f(B-\{g(\beta)\mid\beta\in\alpha\})\in B$，得到

矛盾。

(ii)⇒(iii)：假设$\langle A, <_A\rangle$是一个(非空)偏序，其中每条链都有上界。需要证明A有极大元。令$<$为由(ii)保证的A的良序。如下找出序$<_A$下的一个极大元：超穷归纳定义序数上的函数g，使得，若存在A的$<$-极小元，则$g(\alpha)$是A的$<$-极小元x，满足对每个$\beta<\alpha$,有$g(\beta)<_A x$，否则为某个固定的$z\notin A$。如前一种情况，存在最小的α，满足$g(\alpha)=z$且$g\upharpoonright\alpha$是1-1的。显然，(根据超穷归纳)如果$\beta<\gamma<\alpha$，那么$g(\beta)<_A g(\gamma)$且$\{g(\beta)\mid\beta<\alpha\}$是$A$中一条链。当然，$g(0)\in A$，因为它非空，从而$\alpha\neq 0$。如果$\alpha$是极限序数，那么由每条链都有一个上界的假设，将存在一个$x\in A$，满足对所有$\beta<\alpha$，有$g(\beta)<_A x$，从而$g(\alpha)$就在A中，与对α的选择矛盾。因此，α是一个后继序数$\beta+1$且$g(\beta)$必为A的一个极大元，否则根据定义，$g(\alpha)$将在A中。

(iii)⇒(i)：令A为一个非空集合的集合。令B是函数g的族，其中g的定义域为A的某个子集，该子集由满足$g(b)\in b$的那些b构成。令$<$为B上由函数延拓给出的序($g<h$，当且仅当$\mathrm{dom}\,g\subset\mathrm{dom}\,h$且对每个$x\in\mathrm{dom}\,g$，有$g(x)=h(x)$)。显然，若$C\subseteq B$是一条链，则$\bigcup C\in B$且它是$C$的一个上界。因此，由(iii)存在$B$的一个极大元$g$。如果存在某个$x\in A$不在$g$的定义域中，那么可以通过把任何$z\in x$加入配对$\langle x, z\rangle$来延拓$g$，这与$g$在$B$中的极大性矛盾。因而$g$在整个$A$上有定义，所以它就是我们所要的选择函数。 □

我们在习题1~4中给出选择公理的一些其他常见的等价形式。

作为超穷递归定义的第二个应用，(对所有序数α)定义集合的分层$R(\alpha)$，它由幂集运算迭代产生，这种运算定义了所谓的有秩全域R。

定义 10.3 通过超穷归纳定义序数上的一个函数算子$R(\alpha)$，满足$R(0)=\varnothing$，$R(\alpha+1)=P(R(\alpha))$且对极限序数λ，有$R(\lambda)=\bigcup\{R(\alpha)\mid\alpha<\lambda\}$。一个集合$A$的**秩**(rank)为$\alpha$，是指$A\in R(\alpha+1)$且对所有$\beta\leqslant\alpha$，有$A\notin R(\beta)$，即$\alpha$是满足$A\in R(\alpha+1)$的最小序数。记$A$的秩为$\rho(A)$。用$R$表示所有有秩集合(即那些以某个$\alpha$为秩的集合)的族。

由于在序数定义中包含了良基性，我们对其性质的探讨，包括通过超穷归纳来定义这些函数性质的能力，都不依赖于基础公理。更进一步地，由于ON的良基性，R也是良基的：

引理 10.4 若$y\in x\in R$，则$y\in R$且$\rho(y)<\rho(x)$。

证明 假设$\rho(x)=\alpha$，从而$x\in P(R(\alpha))$。因此，$y\in R(\alpha)$，这样y就是有秩的。此外，若$\alpha=\beta+1$，则$\rho(y)\leqslant\beta<\alpha$，若$\alpha$是极限序数，则由定义，$y\in R(\beta)$对某个$\beta<\alpha$成立，因此，也有$\rho(y)\leqslant\beta+1<\alpha$。 □

定理 10.5 在ZF中不用基础公理可以证明R是良基的，即每个有秩集合的集合A有$\in$-极小元。

证明 给定一个集合A，满足每个$x\in A$都有一个秩，考虑集合$B=\{\alpha\mid\alpha$是某个$x\in A$的秩$\}$。由于B是一个序数的集合，它有一个最小元α。令x为A中一个秩为α的元素。我们断言$x\cap A=\varnothing$。否则，就有某个$y\in x\cap A$。但根据引理10.4，y的秩小于x的秩，与我们对α和x的选择矛盾。 □

现在的目标是在ZF中用基础公理证明相反的结论，即每个集合都是有秩的。首先介绍非常有用的传递闭包的概念。

引理 10.6 对每个集合x都存在一个最小的传递集合y，满足$x\in y$。这个集合称为x的**传递闭包**并记为$\mathrm{TC}(x)$。

证明 通过(在自然数上)递归定义一个函数f，使得$f(0)=x$且$f(n+1)=\cup f(n)\cup f(n)$。令$y=\cup\{f(n)\mid n\in\omega\}$。由于$x\in f(0)$，$x\in y$。为证明$y$是传递的，考虑任何$u\in v\in y$。根据$y$的定义，$v\in f(n)$对某个$n$成立，所以$u\in f(n+1)$。因此，$u\in y$。最后，若$x\in z$且$z$是传递的，由归纳立即得$f(n)\subset y$对每个$n\in\omega$成立，所以$y$确实是含有$x$的最小的传递集合。 □

定理 10.7 *对每个集合x都存在一个序数α，满足$x\in R(\alpha)$。*

证明 如果存在一个集合$x\notin R$，那么应用基础公理于集合$\{z\in \mathrm{TC}(x)\mid z\notin R\}$，则存在一个集合$A$，满足$A\notin R$且每个$x\in A$都是有秩的。根据替换公理，函数性质$\rho$在$A$上的值域是一个集合$B=\{\rho(x)\mid x\in A\}$。由于$B$是一个序数的集合，$\cup B$是一个序数$\beta$，它至少和$B$的每个元素一样大(命题9.3(iii))。根据$\rho$的定义和习题5，$x\in R(\beta+1)$对每个$x\in A$成立，所以$A\in R(\beta+2)$是有秩的。 □

有了基础公理，每个集合都由序数上的幂集运算迭代逐步构造出来。康托称幂集运算为第二构造原理。第一构造原理是构造序数的。这解释了康托认为他的两个构造原理是集合论基础的原因。以他的观点来看，用这些原理通过有秩全域的构造就建立了所有的集合。

没有基础公理，或许会存在其他集合，但(至少在数学中)我们不需要它们。用不用选择公理，都可以不用基础公理在 ZF 中(习题7)证明 ZF 中每条公理(包括选择公理，如果在一开始假设了的话)在有秩全域R中都成立。因此，可以得到一个满足基础公理的集合结构，在其中所有不需要基础公理的数学都能很好的发展。

习题

证明：下面 1 ~ 4 中每条结论都和选择公理等价。

1. 如果$\langle A,\leqslant\rangle$是一个偏序集，那么$A$的每条链都包含在一条极大链中。(一条链$C$是$A$的一个线性序子集；它是极大的，是指对所有$a\in A-C$，$C\cup\{\alpha\}$都不是链。)
2. 如果A是一个(集合的)集合，以包含关系为偏序，满足对每条非空链$C\subseteq A$，$\cup C\in A$，那么A有极大元。等价地，如果对每个这样的C，$\cap C\in A$，那么A有极小元。
3. 对每个集合A，若它由互不相交的(即若x，$y\in A$，则$x\cap y=\varnothing$)非空集合组成，则存在一个集合C，满足对所有$y\in A$，$C\cap y$只有一个元。(这样一个C称为A的一个选择集(choice set)。)
4. 一个由非空集合构成的集合的笛卡儿积是非空的。(如果I是一个非空集合且f是一个函数，满足对每个$i\in I$，$f(i)$是一个非空集合，那么集合族$\{f(i)\mid i\in I\}$的笛卡儿积是所有定义域为I且对每个$i\in I$，满足$g(i)\in f(i)$的函数g的一个集合。)
5. (对α归纳)证明：$\alpha\leqslant\beta\Rightarrow R(\alpha)\subseteq R(\beta)$。
6. 证明：$\langle R(\omega\cdot 2),\in\rangle$是所有策梅洛公理(公理1 ~ 7)的一个模型，但不是替换公理的模型。
7. 对 ZFC 的每条公理ϕ，(在 ZFC 中)证明：ϕ在R中成立。(提示：对关于公式$\psi(x,y)$的替换公理实例(在R中解释)，考虑这样的函数，它把x映为某个集合的秩，该集合中的元素是使得$\psi(x,y)$在R中成立的那些y。此时，应用 ZFC 中的替换公理。)
8. 证明：引理 10.1 的证明中所定义的序$\langle B,S\rangle$是一个良序。
9. 证明：如果P是集合的一个性质，它满足存在一个x，$P(x)$成立，那么就存在一个$\in$-极小的x，$P(x)$成立，即存在一个x满足$P(x)$成立但不存在$y\in x$使得$P(y)$成立。(提示：考虑使$P(x)$成立的一个x的传递闭包。)

第十一节 基数和基数算术

这一节把从第七节开始的对于集合大小的研究拓展到不可数集上，以此结束对于集合论的介绍。我们称两个集合A和B是有相同大小或是等势的($|A|=|B|$)，若存在一个双

射f: $A \to B$。这是一个等价关系(定理7.2)，但是证明中我们遗留了选择等价关系的代表元的问题。好，现在就让我们来解决这个问题。假设选择公理成立，那么每个集都可良序化并且由第八节习题4知道每个集都和某个序数有相同的势。由第八节习题2这个序数是唯一决定的。很自然的对每个给定的等价类我们选取最小的这样的序数作为这个等价类的典型代表。

定义 11.1 一个序数κ是一个**基数**(cardinal)，如果不存在序数$\alpha<\kappa$，使得$|\kappa| \equiv |\alpha|$。

当然，基数作为序数的一个子集仍然在$\in$下为良序。

定理 11.2 在ZFC中能够证明：对任何集合A，存在唯一的基数κ，使得$|A| \equiv |\kappa|$。我们把这个关系称为A的势是κ，记作$|A| = \kappa$。

注意到，和序数一样，基数上的序$\in$，与第七节中按照集合之间的一一映射所给出的序相同。证明留作习题2~3。

当然，(根据定理7.4)第一批基数是有穷序数0，1，2，…。于是第一批无穷基数是第一个无穷序数ω，我们称基数为ω的集合为可数的。当将ω这个集合当作基数来考虑时，一般记为$\aleph_0$，这是康托给出的。$\aleph$(aleph，阿列夫)是希伯来字母表的第一个字母，康托用之来表示无穷基数。由康托定理(定理7.13)，2^ω是不可数的，并且根据定理11.2，存在一个不可数基数。在康托的记号中记作$\aleph_1$。第七节习题11中给出的一般形式的康托定理表明：对任何基数κ，$\kappa < |2^\kappa|$，因而不存在最大的基数。(见定理11.8和定理11.14中这一事实的其他版本。)应用这个事实，可以用超穷递归来定义无穷基数的一个枚举。

定义 11.3 通过归纳在序数上定义一个函数性质F如下：$F(0) = \omega$并且$F(\alpha)$是最小的基数κ，对每个$\beta<\alpha$，满足$F(\beta) < \kappa$。$F(\alpha)$通常记作$\aleph_\alpha$。

于是，$\aleph_\alpha$列举了所有的基数。注意到，当λ是一个极限序数时，$\aleph_\lambda = \bigcup\{\aleph_\alpha \mid \alpha < \lambda\}$(习题4)。

现在可以将第七节中定义在有穷基数和$\aleph_0$上的常规的算术函数拓展到所有基数上。类似于有穷基数上的定义那样(见第七节习题5~9)，加法对应于无交并，乘法对应于笛卡儿积，指数运算对应于某个适当的函数集。

定义 11.4(基数的算术) 令κ，λ为基数。定义基数的**加法**、**乘法**和**指数运算**如下：

1. 加法：$\kappa + \lambda = |(\kappa \times \{0\} \cup (\lambda \times \{1\})|$。
2. 乘法：$\kappa \cdot \lambda = |\kappa \times \lambda|$。
3. 指数运算：$\kappa^\lambda = |{}^\lambda\kappa|$(${}^\lambda\kappa$是由所有定义域为$\lambda$值域包含在$\kappa$中的函数构成的集合)。

我们先举出一些由这些运算的定义可以立即得出的简单事实，然后再完整地描述无穷基数加法和乘法可得到的结果。我们会看到，这些结果本质上都是平凡的。

命题 11.5 令κ，λ，μ为基数，则

(i)$\kappa + 0 = \kappa$，$\kappa + \lambda = \lambda + \kappa$，$\kappa + (\lambda + \mu) = (\kappa + \lambda) + \mu$

(ii)$\kappa \cdot 1 = \kappa$，$\kappa \cdot \lambda = \lambda \cdot \kappa$，$\kappa \cdot (\lambda \cdot \mu) = (\kappa \cdot \lambda) \cdot \mu$

(iii)$(\kappa \cdot \lambda)^\mu = \kappa^\mu \cdot \lambda^\mu$，$(\kappa^\lambda)^\mu = \kappa^{\lambda \cdot \mu}$

证明 这些都是定义的简单推论，留作习题5。 □

将1加到一个无穷基数上是很容易计算的。

命题 11.6 如果κ是一个无穷基数，则$\kappa + 1 = \kappa$。

证明 由基数的定义，只需定义一个双射f: $\kappa + 1 \to \kappa$，其中$\kappa + 1 = (\kappa \times \{0\}) \cup (\{0\} \times$

$\{1\})=(\kappa\times\{0\})\cup\{\langle 0,1\rangle\}$。令 $f(\langle 0,1\rangle)=0$，对 $n\in\omega$，令 $f(\langle n,0\rangle)=n+1$ 且对 $\omega\leqslant\alpha<\kappa$，令 $f(\langle\alpha,0\rangle)=\alpha$。则此函数显然就是我们想要得到的双射。 □

关于无穷基数加法和乘法的所有其他情形都可用下面的定理来确定：

定理 11.7 *如果 κ 和 λ 是基数，其中至少一个是无穷且另一个大于零，则 $\kappa+\lambda=\kappa\cdot\lambda=\max\{\kappa,\lambda\}$。*

证明 根据命题 11.5，加法和乘法是可交换的。不失一般性，可以假设 $\kappa=\max\{\kappa,\lambda\}$，因而 κ 是无穷的。如果 $\lambda=1$，定理可以由命题 11.5 和命题 11.6 推出。另一方面，如果 $\lambda\geqslant 2$，由定义可以得到 $\kappa\leqslant\kappa+\lambda\leqslant\kappa\cdot\lambda\leqslant\kappa\cdot\kappa$。于是只需证明，对所有无穷基数 κ，$\kappa\cdot\kappa\leqslant\kappa$。通过对 κ 归纳来证明这个事实。基本的情形 $\kappa=\aleph_0$ 可由定理 7.10(ii)得到。

现在假设 $\beta\cdot\beta\leqslant\beta$ 对所有的基数 $\beta<\kappa$ 成立。在 $\kappa\times\kappa$ 上定义如下一个序 $<_1$：$\langle a,b\rangle<_1\langle c,d\rangle$，当且仅当 $\max\{a,b\}<\max\{c,d\}$；或 $\max\{a,b\}=\max\{c,d\}$ 且 $a<c$；或 $\max\{a,b\}=\max\{c,d\}$，$a=c$ 且 $b<d$。立即可以得到 $\langle\kappa\times\kappa,\leqslant_1\rangle$ 是一个良序（习题 6），因而（根据第八节习题 4）这个序与某个序数 δ 同构。显然，这个序的每个真初始段（作为一个集）对某个 $\gamma<\kappa$，是包含在 $\gamma\times\gamma$ 里的。当然，$|\gamma|<\kappa$，因为 κ 是一个基数。于是，由归纳可得，每个 $<_1$ 的真初始段的势都小于 κ。于是每一个 δ 的元素都是小于 κ 的，因而，$\delta\leqslant\kappa$ 满足要求。 □

如果没有选择公理，基数算术将会变得十分复杂。事实上，由一个一一的函数 $f: A\to B$ 的存在性定义出的序 $|A|\leqslant|B|$ 甚至不是一个线性序。可以找到一个无穷集 A（即对任意的 $n\in\omega$，不存在双射 $f: A\to n$，因而也不存在这样的单射），使得不存在单射 $g:\omega\to A$（当然更不存在 A 到 ω 内的了）。我们仍然能够证明的重要事实仅剩下施罗德–伯恩斯坦(Schröder-Bernstein)定理和哈托(Hartog)定理。

定理 11.8

1. *(施罗德–伯恩斯坦定理)如果存在两个一一函数 $f: A\to B$ 和 $g: B\to A$，则存在双射 $h: A\to B$。也就是说，如果 $|A|\leqslant|B|$ 且 $|B|\leqslant|A|$，则 $|A|=|B|$。*

2. *(哈托定理)对每一个集合 A，存在一个良序集 B，从而有一个序数 κ，使得不存在从 B(或 κ)到 A 的一一映射。*

证明 留作习题 7。 □

就算使用选择公理，基数的指数运算也是远非平凡的。已经证明了(定理 7.13) $\omega=\aleph_0<2^{\aleph_0}=|P(\omega)|$。通过相同的对角线方法，康托的这个重要定理在一般情况下也是对的。

定理 11.9(康托) *对每一个基数 κ，$\kappa<2^\kappa=|\mathcal{P}(\kappa)|$。*

证明 定理由第七节习题 11 可以立即得到。也可以由命题 11.10 和定理 11.14 得出。 □

我们已经知道，只要知道基数的幂集的势，即 $|2^\kappa|$，就足以计算出几乎所有的关于无穷基数的指数函数的势。例如，有下面的计算：

命题 11.10 *如果 λ 和 κ 是两个基数，且 $1<\lambda\leqslant\kappa$，则 $\lambda^\kappa=2^\kappa$。*

证明 显然 $2^\kappa\leqslant\lambda^\kappa\leqslant\kappa^\kappa$。另一方面，由 $\kappa<2^\kappa$，$\kappa^\kappa\leqslant(2^\kappa)^\kappa=2^{\kappa\cdot\kappa}=2^\kappa$（根据命题 11.5(ii)和定理 11.7）。因此，$2^\kappa=\lambda^\kappa=\kappa^\kappa$，命题得证。 □

现在我们只知道 $2^\kappa>\kappa$。一个自然的猜想是：κ 的幂集是尽可能小的，即对每一个基数 $\aleph_\alpha$，有 $2^{\aleph_\alpha}=\aleph_{\alpha+1}$。因为 $2^{\aleph_0}$也是连续统(the continuum)的势，也就是实数集合的势(习题8~9)，所以这个猜想的第一个例子通常被称为(康托的)连续统假设(Continuum Hypothesis, CH)：$2^{\aleph_0}=\aleph_1$。一般形式被称为广义连续统假设(Generalized Continuum Hypothesis, GCH)：对每一个序数 α，有 $2^{\aleph_\alpha}=\aleph_{\alpha+1}$。连续统假设困扰了康托的最后一段岁月，并且在几十年内都没有得到解决。即使是现在，我们也只知道：现有的集合论公理，比如 ZFC(以及实际上每个被大家所接受的 ZFC 的扩张)，不足以决定连续统的势。

定理 11.11(Gödel[1938, 1939, 2.3]和[1940, 3.3]) *如果 ZF 是相容的，则 ZFC 加上 CH 是相容的，所以在 ZFC 中不可能证明 CH 不成立。*

定理 11.12(Cohen[1963, 1964 和 1966, 3.3]) *如果 ZF 是相容的，则 ZFC 加上 CH 的否定是相容的，所以在 ZFC 中不可能证明 CH 成立。*

事实上，在科恩的工作基础之上继续进行的研究表明：连续统可以取到几乎任何的值。完全确定一般的 2^κ和特殊的 2^ω的值的可能限制，以及计算指数函数的所有其他的值(κ^λ，其中 $\lambda<\kappa$)，(除了函数 2^κ 的实际值之外)还依赖于一个共尾度概念和 κ 与其共尾度之间的关系。

定义 11.13 *一个基数 κ 的***共尾度** *$cf(\kappa)$，是最小的基数 λ，使得存在一个函数 f: $\lambda\to\kappa$，它的值域在 κ 中是无界的。一个基数 κ 是***正则的***(regular)，如果 $cf(\kappa)=\kappa$；否则是***奇异的***(singular)。*

由定义可以立即得出，如果 κ 是正则的且 $\lambda<\kappa$，那么 $\kappa^\lambda=\bigcup\{\beta^\lambda\mid\beta<\kappa\}$。因此，如果知道指数函数 κ^λ 在 κ 奇异且 $\lambda<\kappa$ 的情况下的取值，则对所有的 κ 和 λ，可以由函数 2^γ 的值计算出指数函数的值。事实上，为了对每一个 α 和 β 计算 $\aleph_\alpha^{\aleph_\beta}$，只需要知道 $\kappa^{cf(\kappa)}$的值(习题12)。关于 $\kappa^{cf(\kappa)}$在奇异基数 κ 时的值，有一个经典的事实，这一事实也给出了对连续统取值时唯一可能的限制。这可由康托证明 $2^\kappa>\kappa$ 所用的最初的对角线方法做一些改变来证明。

定理 11.14(库尼西定理) *对所有的基数 κ，$\kappa^{cf(\kappa)}>\kappa$。*

证明 令 $\lambda=cf(\kappa)$，取定一个函数 f: $\lambda\to\kappa$ 使其值域无界。考虑任何一个函数 g: $\kappa\to\kappa^\lambda$。证明 g 不是一个满射，从而 $\kappa<\kappa^\lambda$。令 h: $cf(\kappa)\to\kappa$ 定义为：使得 $h(\alpha)$ 是不在 $\{g(\beta)(\alpha)\mid\beta<f(a)\}$中的 κ 的最小元素。(注意：根据共尾度的定义，h 在 λ 上有定义。)我们断言：正如所要求的那样，$h\notin rg(g)$。否则，存在某个 $\beta<\kappa$，使得 $h=g(\beta)$。现在，根据对 f 的选择，对某个 $\alpha<\lambda$，有 $\beta<f(\alpha)$，因而，$h(\alpha)=g(\beta)(\alpha)$，但是根据定义，对每个$\beta<f(\alpha)$，有 $h(\alpha)$与 $g(\beta)(\alpha)$不等，从而矛盾。 □

推论 11.15 $cf(2^{\aleph_0})>\aleph_0$。

证明 如果 $cf(2^{\aleph_0})=\aleph_0$，则根据库尼西定理，$(2^{\aleph_0})^{\aleph_0}>2^{\aleph_0}$，但是，(根据命题 11.5 和定理 11.7)有$(2^{\aleph_0})^{\aleph_0}=2^{\aleph_0\times\aleph_0}=2^{\aleph_0}$，从而得到矛盾。 □

这个推论给出了连续统取值的唯一可能的限制。本质上，如果存在一个 ZFC 的模型，其中 κ 是一个共尾度大于 ω 的基数，则存在一个 ZFC 的模型，使得连续统的势在其中恰好是 κ。事实上，对正则的 κ，2^κ 的势的唯一可能的限制是由库尼西定理、康托定理以及 $\gamma\leqslant\kappa$ 从而 $2^\gamma\leqslant2^\kappa$ 这一显然限制所决定的(Solovay[1963, 1965, 3.3]和 Easton[1964, 3.3])。在很长一段时间里，人们相信基数的指数运算没有其他的限制，即当 κ 是奇异基数时，2^κ 也

可以取任何与库尼西定理相容的值。这个观点被 Silver[1974, 3.3]否定了，他证明了：如果$\aleph_\alpha$是有不可数共尾度的奇异基数，并且对$\beta<\alpha$，有$2^{\aleph_\beta}=\aleph_{\beta+1}$，则$2^{\aleph_\alpha}=\aleph_{\alpha+1}$。从这之后，一套 ZFC 中关于奇异基数的完整的算术理论发展起来，其中最重要的是谢拉赫(Saharon Shelah)[1994, 3.3]的工作。下面的结论是这些结论中最重要的一条：

定理 11.16(谢拉赫定理)　如果$2^{\aleph_0}<\aleph_\omega$，则$(\aleph_\omega)^{\aleph_0}<\aleph_{\aleph_4}$。

这个结论中，$\aleph_4$所起的特殊作用还未被完全搞清，很可能还会有进一步的发展。

习题

1. 证明：如果α是一个序数且$A\subseteq\alpha$，则对某个序数$\mu\leq\alpha$，有$\langle A,\in\rangle\sim\mu$。(提示：定义一个函数$f$，使得对每个$\gamma<\alpha$，有$f(\gamma)=\{f(\beta)\mid\beta<\gamma\}$。)

2. 证明：如果κ和λ是基数，则$\lambda\leq\kappa$，即$\lambda\in\kappa$或$\lambda=\kappa$，当且仅当存在一个单射$f:\lambda\to\kappa$。

3. 证明：如果κ和λ是基数，则$\lambda=\kappa$，当且仅当存在一个双射$f:\lambda\to\kappa$，当且仅当存在两个单射$g:\lambda\to\kappa$和$h:\kappa\to\lambda$。

4. 证明：如果λ是极限序数，则$\aleph_\lambda=\bigcup\{\aleph_\alpha\mid\alpha<\lambda\}$。

5. 证明命题 11.5。

6. 证明：定理 11.7 中定义的序$<_1$是一个良序。

7. 证明定理 11.8：施罗德-伯恩斯坦定理和哈托定理。(提示：对第一个，用定理中的符号，令$A_0=A$，$B_0=B$，$A_{n+1}=\{g(x)\mid x\in B_n\}$，$B_{n+1}=\{f(x)\mid x\in A_n\}$，$A_\omega=\bigcap\{A_n\mid n\in\omega\}$，$B_\omega=\bigcap\{B_n\mid n\in\omega\}$。令$h(x)=f(x)$，如果$x\in A_\omega\cup(\bigcup\{A_{2n}-A_{2n+1}\mid n\in\omega\})$；否则，令$h(x)=g^{-1}(x)$。证明$h$就是我们所需的从$A$到$B$的双射。对第二个，考虑$A$的子集上所有的良序构成的集合和所有与这些良序相似的序数所构成的集合。令κ大于这个集中的所有元素。)

8. 证明：实数集$\mathcal{R}$、闭的单位实数区间$[0,1]$和开的单位实数区间$(0,1)$具有相同的势。

9. 通过“无穷基小数”来证明ω^ω和$[0,1]$等势：给定一个函数$f:\omega\to\omega$，令$s(n)=f(0)+f(1)+\cdots+f(n)$。定义函数$F:\omega^\omega\to[0,1]$为$F(f)=\sum\frac{2^{f(n)-1}}{2^{s(n)}}$，证明：$F$是既单又满的。

10. 证明：从实数到实数上的连续函数构成的集也具有势$2^{\aleph_0}$。

11. 假设 GCH，计算下列基数：$\aleph_3{}^{\aleph_6}$，$\aleph_5{}^{\aleph_4}$，$\aleph_\omega{}^{\aleph_1}$，$\aleph_\omega{}^{\aleph_0}$以及对每个$\alpha$，计算：$|R(\alpha)|$。

12. 假设对每一个基数$\aleph$都知道$\aleph^{cf(\aleph)}$的值，对每一个α和β，说明计算$(\aleph_\alpha)^{\aleph_\beta}$的方法。(提示：首先计算函数$2^{\aleph_\alpha}$：如果$\aleph_\alpha$是正则的，那么$\aleph_\alpha{}^{cf(\aleph_\alpha)}=2^{\aleph_\alpha}$。如果$\aleph_\alpha$是奇异的，且对$\gamma<\alpha$，$2^{\aleph_\gamma}$最终是常数，那么$2^{\aleph_\alpha}$是$2^{\aleph_\gamma}$的极限值。否则，$2^{\aleph_\alpha}=\lambda^{cf(\lambda)}$，其中$\lambda=\bigcup\{2^{\aleph_\gamma}\mid\gamma<\alpha\}$。然后证明$(\aleph_\alpha)^{\aleph_\beta}$的值可归纳地由所有$2^{\aleph_\gamma}$和$\aleph_\gamma{}^{cf(\aleph_\gamma)}$的变元值决定。关键的事实是当$cf(\aleph_\alpha)\leq\aleph_\beta<\aleph_\alpha$，并且$\aleph_\alpha=\bigcup\{(\aleph_\gamma)^{\aleph_\beta}\mid\gamma<\alpha\}$时，有$\aleph_\alpha^{\aleph_\beta}=\aleph_\alpha{}^{cf(\aleph_\alpha)}$。)

进一步阅读建议

Fraenkel 和 Bar-Hillel[1958, 3.3]和[1973, 3.3]是一本经典基础教材，它包括了大量对集合论起源的探讨。另一本标准教材是 Levy[1979, 3.3]。更多最近的本科生教材包括 Devlin[1993, 3.3]、Moschovakis[1994, 3.3]和 Vaught[1994, 3.3]。Quine[1969, 3.3]从哲学(主要是本体论)角度出发讨论基本的集合论观点，并表述了与 ZFC 完全不同的集合论公理体系(新基础论)。

科恩(Cohen)关于选择公理和连续统假设独立于 ZFC 的原稿[1966, 3.3]，至今还是一篇有价值的展示哥德尔的可构成集域和科恩力迫方法基本内容的文章。现在关于这些内容(还有更多内容)的标准的研究生基础教材是 Kunen[1980, 3.3]，十分具有可读性。Jech

[1978, 3.3]也是一本很重要的研究生教材，他采用一种利用布尔值模型来研究力迫的不同的方法。关于选择公理许多变形及相关原理更深入的论述可在 Jech[1978, 3.3]以及 Rubin 和 Rubin[1963, 3.3]，[1985, 3.3]中找到。

Kanamori[1994, 3.3]是最近一本处理许多其他主题包括大基数、描述集合论和确定性公理的高级教材。这本书有一个涵盖很广的参考文献，可以作为本学科近期发展的历史指导。Kanamori[1996, 2.2]是一篇极好地描述了直到科恩之前的集合论历史的短文。关于近期基数算术的工作的一篇好的综述(其中强调了谢拉赫的工作)，可以在 Jech[1995, 3.3]中找到。

附录A 历史回顾

我们从类比微积分和数理逻辑的历史开始。

第一节 微 积 分

在古希腊黄金时代的雅典，柏拉图(Plato，公元前428—348)曾经把懂得几何作为进入他的哲学学院的先决条件，因为一个很好地掌握了几何的人也必定是一个能够进行正确和严格推理的人(见 Thomas[1939，1.1])。欧几里得(公元前300—)强调了公理化方法的重要性，即从公理开始进行演绎。(从这个观点来看，逻辑就是研究演绎的学问。)欧几里得和阿基米德(Archimedes，公元前287—212)以及他们的前辈说明了如何用综合几何学来计算许多简单图形与立体图形的面积和体积。他们也说明了如何用几何学来解决许多力学、流体静力学和几何光学的简单问题。从欧几里得和阿基米德到莱布尼茨(Leibniz，1640—1710)和牛顿(Newton，1640—1722)的20个世纪里，人们用欧几里得和阿基米德几何学中的特殊方法解决了一个又一个越来越困难的计算面积和体积的问题以及流体力学的问题。用这种几何方法得到的物理学或者数学上的每一个进步，都需要有伽利略(1564—1642)或惠更斯(1629—1695)那样的非凡数学天赋。在笛卡儿的发现之后事情有了根本性的转变，作为其著作《Discours de la Methode》[1637，2.3]的附录发表出来的文章，使得人们能够把几何问题转化为等价的代数问题。几何的方法被代数的计算取代了。

在费马(Fermat，1601—1665)的工作和牛顿的老师巴罗(Barrow，1630—1677)以及莱布尼茨的前辈卡瓦列里(Cavalieri，1598—1647)的工作中，已经显示出积分和微分的符号——代数方法的很强的迹象了。牛顿和莱布尼茨发现的微分和积分的符号方法，使得即使不是数学天才的后辈们也能用普通的微积分来发展科学和工程了。这些方法至今仍是理解、模拟、模仿、设计和发展物理和工程系统的基础。

第二节 逻 辑

亚里士多德(公元前384—322)三段论的理论也可以追溯到古希腊的黄金时代和柏拉图学院的辩论(见柏拉图的《Euthydemus》)。这可以在其被古代的编辑称为《Organon》的著作集中找到。它由《Catagoriae》、《De Interpretatione》、《Analytica Priora》、《Analytica Posteriora》、《Topica》和《De Sophisticis Elenchis》等部分组成。我们仅仅讨论《Analytica Priora》中三段论的原理。

这只是第一次成功地使用“所有”和“某一个”进行推理的演算。在现代的术语中，我们把“所有”和“某一个”翻译为量词“对所有”和“存在”。在现代人的眼中，三段论以它名词性的表达、术语的通用性和特殊性而显得奇特。但它有实实在在的动机。对亚里士多德而言，世界是由具有给定性质 P 或不具有性质 P 的对象 c 组成的。在现代的符号里，字母 P 被称为谓词符号。P 的一个特殊解释是由指定的非空定义域 C 及其记为 P 的子集给出的。于是，当 x 是一个在 C 中取值的变元时，$P(x)$ 就是一个读作“x 具有性质 P”的逻辑公式。如果 c 是一个特定对象的名称，$P(c)$ 就是读作“c 具有性质 P”的逻辑公式。

现在一个对象可能同时具有很多种不同的性质。一个对象 c 可能同时是硬的、圆的、红的、比水轻、在这个房间里、在地板上、在东南角。在 17 世纪末，莱布尼茨认为对象应该能通过知道它所有的性质来唯一地描述。莱布尼茨把这一想法称为*不可辨别者的同一性原理*(the principle of identity of indiscernibles)。

从一个对象 c 具有几种性质来推演出它具有另外一种性质就是亚里士多德的三段论所要陈述的问题。逻辑的这一用法尤其是林奈(Linnaeus，1707—1788)的生物学分类的特征，包括它的属、种和品种。他的体系是亚里士多德生物学直接的理性派生物。因此常常称亚里士多德为“生物学之父”。他的生物学概念和三段论的概念是紧密相联的。从中世纪到 1900 年，三段论一直是标准的大学课程里逻辑学、修辞学和语法学三门课程的一部分。即使已经被现代的数理逻辑所淘汰，在很多天主教大学里，它作为逻辑推理的主要训练内容，仍然没有丝毫减少和改变。

三段论的主要作用是检查量词“对所有的 x”和“存在一个 x”在论证中是否被正确使用。目的是排除那些看似使用了正确逻辑原理其实则不然的错误论证。在书写完全作为推理规则的三段论时，我们效仿亚里士多德的追随者克利西普斯(Chryssipus，公元前 207—)，把三段论完全写成各种推理规则。“Barbara 模式”是这种形式三段论的永真模式的一个例子。

从 “所有 P 是 Q”
和 “所有 Q 是 R”
推出 “所有 P 是 R”。

在现代的逻辑符号中，分别用 $P(x)$ 表示“x 是一个 P”，$Q(x)$ 表示“x 是一个 Q”，$R(x)$ 表示“x 是一个 R”，$(\forall x)$ 表示“对所有的 x”，$(\exists x)$ 表示“存在一个 x”，“$\rightarrow$”表示“蕴涵”。翻译成现代的推理规则，Barbara 模式就变成了

从 $(\forall x)(P(x)\rightarrow Q(x))$
和 $(\forall x)(Q(x)\rightarrow R(x))$
推出 $(\forall x)(P(x)\rightarrow R(x))$。

在这种符号中，P，Q，R 称作一元谓词(或关系)符号。一个非永真的亚里士多德模式是

从 “某些 P 是 Q”
和 “某些 Q 是 R”
推出 “某些 P 是 R”。

在现代的符号中，用$\wedge$代替“且”，这一“规则”被翻译成如下形式：

从 $(\exists x)(P(x)\wedge Q(x))$
和 $(\exists x)(Q(x)\wedge R(x))$
推出 $(\exists x)(P(x)\wedge R(x))$。

(给出这一模式的一个反例！)

三段论处理四种形式的命题，称为范畴命题，它们中世纪的名称是 A、E、I 和 O：

A) “每一个 P 都是一个 Q”
E) “没有 P 是一个 Q”
I) “某些 P 是一个 Q”
O) “某些 P 不是一个 Q”。

永真的模式都会根据命题序列给出助记名称。因此上面列出的永真模式的两个假设和一个结论构成序列 AAA，被称为“Barbara”。

考虑助记名为“Celarent”的序列 EAE：

从　　“没有 Q 是 R”

和　　“每一个 P 都是 Q”

推出　　“没有 P 是 R”。

现代符号里用$\neg$表示“不是”，所以这个模式可以翻译成如下的现代符号：

从　　$\neg(\exists x)(Q(x)\wedge R(x))$

和　　$(\forall x)(P(x)\to Q(x))$

推出　　$\neg(\exists x)(P(x)\wedge R(x))$。

习题

从中世纪记忆性名称对应的元音字母出发，重新构造出有两个前件和一个结论的三段论，并把它们翻译成现代符号。

1. Darii　2. Ferio　3. Cesare　4. Camestres　5. Festino　6. Baroco　7. Darapti　8. Disamis　9. Datisi　10. Felapton　11. Bocardo　12. Ferison　13. Bramantip　14. Camenes　15. Dimaris　16. Fesapo　17. Fresison。

(提醒：亚里士多德假设谓词是非空的。这意味着根据现代的惯例，要保证 7、10、13 和 16 的有效性，这些假设必须明确地给出。)

亚里士多德给出了从其他模式到这些模式的系统的推导。他的工作给出了第一个公理系统，用于从一些逻辑真理推导出另外一些逻辑真理。另外，他还构造了错误模式的反例和否定命题(矛盾命题)的规则。亚里士多德也是第一个基于联结词$\Diamond p$(“有可能是 p”)和$\Box p$(“一定是 p”)，对模态逻辑(第四章的内容)进行系统讨论的人。这些联结词和上面提到的那些不同，它们不是“真值泛函”。这就是说 p 的真假并不能决定$\Box p$ 和$\Diamond p$ 的真假：p 也许是可能的和错的，或是可能的和对的。(关于古希腊逻辑学家工作的更详细分析，见：Lukasiewicz[1957，2.2]、Bochenski[1951，2.2]和 Mates[1961，2.2]。)

尽管三段论有助于澄清哲学讨论，但它对数学家并没有实质的影响。在亚里士多德之前，数学家就非常严密地进行推理了。事实上，他们的工作是精确推理的传统模型。然而，他们的推理并没有完全被三段论所描述。什么被遗漏了呢？后来才知道，一个简短的答案：命题逻辑的其余部分和一个有争议的关系的概念。从麦加拉学派的斐洛(Philo，公元前300—)的工作发展开去，斯多亚学派的克利西普斯引入了蕴涵(现在记作$\to$)，合取(现在记作$\wedge$)和不相容选言(exclusive disjunction，表示“P 或 Q，但不是两个同时”)。现在，最后提到的那个联结词被相容选言(inclusive disjunction)所替代，“P 或 Q 或两个都是”，记作 $P\vee Q$。克利西普斯理解到了命题逻辑的特性：由这些联结词所表示的复合命题的真假值是由各个部分命题的真假值决定的。

至于关系，欧几里得几何学就是基于关联关系 $R(x, y)$的，也就是说 x 和 y 是相关联的，这里的 x，y 是点、线或面。因此一个点可能和一条线相关联(点在线上)；一条线可能和一个面相关联(线在面上)。亚里士多德所遗漏的是二元关系 $R(x, y)$(例如“x 小于 y”)和三元关系 $S(x, y, z)$(例如“z 是 x 与 y 的和”)等也具有作为基本构造单元的特征。他仅仅使用一元关系或谓词 $P(x)$，例如“x 是红的”。他通常把二元关系 $S(x, y)$，例如“x 是 y 的祖父”，编译为性质 $S_y(x)$，即 x 具有性质：是 y 的祖父。

亚里士多德关于量词的理论没有真正的缺陷。只是在他的逻辑公式中缺少明确表示的命题联结词和多元关系。这一缺陷直到 19 世纪晚期才被人修补，如皮尔斯(C. S. Peirce，1839—1914)、施罗德(E. Schröder，1841—1902)和弗雷格(G. Frege，1848—1925)。

第三节　莱布尼茨的梦想

莱布尼茨第一个断言可能存在描述推理的完全形式逻辑(见 Parkinson[1965，2.2])。他确信他能够发展出一种推理的语言和演算，这将和牛顿—莱布尼茨微分和积分运算一样重

要。他把他的新学科称作“lingua characteristica”(通用语言)和“calculus ratiocinator”(推理演算)。人们的思想“将会从直接思考事物本身中解放出来，并且一切仍将是成功的”(Parkinson[1965，2.2]第105页)。他希望用这些新学科去扩展人们的推理能力，把找出如何从给定前提得出所需要的结论和如何检验给定演绎的正确性等大量的劳动转化为符号计算。换句话说，他认为应该存在一种处理关于这个世界的命题的推理演算，这就好比能够解决统治这个世界的数字方程的牛顿—莱布尼茨微积分一样。

莱布尼茨了解一些命题逻辑和类的演算，但他在这方面的工作一直到 Couturat[1903，2.3]发表之前都基本不为人所知。所以他的想法有预见性但没有带来直接的影响。20世纪最伟大的逻辑学家哥德尔(Kurt Gödel，1906—1978)在“罗素的数理逻辑”(Gödel[1944，2.3])一文中对莱布尼茨的梦想作了如下阐述：

“……如果我们相信他所说的，他已经在很大程度上发展出了这个推理演算只是在等待其种子落入肥沃的土地时再发表。他甚至估计出需要多长时间，他的演算方法才能被一些一流的科学家发展成为‘人类将拥有的、远比任何光学仪器提高视力的作用更大的、提高人类推理能力的新工具’。他确定的时间是5年，并且他宣称他的方法不比他那个时代的数学哲学更难学。而且，他反复提到，尽管他所发展的理论还处于初始阶段，但他所有的数学发现都归功于它，人们可以认为，甚至庞加莱都会承认，这是该理论丰富性的充分证明。”

(庞加莱(Poincaré，1854—1912)是19世纪末20世纪初最著名的法国数学家。他并不认为形式逻辑是数学有用的基础，而是和康托(Cantor，1845—1918)一样认为数学是直接扎根于直觉之中的。)

类代数和命题逻辑直到19世纪中期才由德·摩根(De Morgan，1806—1871)[1847，2.3]和布尔(Boole，1815—1864)[1847，2.3]重新发现，进而发展得更加完整。

第四节 19世纪的逻辑

德·摩根(De Morgan)[1847，2.3]扩展了三段论法，引入了命题联结词和它们的定律并且给出了初步的关系理论。他的朋友布尔是用符号代数方法解决数学问题的专家。布尔在微分和差分方程方面的教科书(重印作 Boole[1959，2.3]和[1970，2.3])是高度代数化和算法化的。它们基于形式算法，用微分算子 D 和差分算子 $\Delta(f)=f(x+1)-f(x)$ 的多项式以及它们的形式逆来分别求解微分和差分方程的。现在让我们用现代符号来看一些布尔在逻辑方面的工作。假设 p，q 是命题，并且考虑如下命题联结词：

1. *析取*，记作$(p \vee q)$并且读作“p 或 q”
2. *合取*，记作$(p \wedge q)$并且读作“p 且 q”
3. *否定*，记作$(\neg p)$并且读作“非 p”。

这些联结词的解释为

1. $(p \vee q)$为真，当且仅当 p 和 q 至少有一个为真
2. $(p \wedge q)$为真，当且仅当 p 和 q 都为真
3. $(\neg p)$为真，当且仅当 p 不真。

为了能够以更高的精确性来分析这些概念，我们采用如第一章定义2.1给出的出现在维特根斯坦(Wittgenstein，1889—1951)(见[1974，2.3])和波斯特(Post，1897—1954)(见 Post[1921，2.3])工作中的有关命题逻辑的定义：如果给定一组原始(原子的)命题字母 p，q，

$r\cdots$,那么有以下命题(归纳的)定义:

(i)原子字母是命题。

(ii)如果 α, β 是命题,那么$(\alpha\vee\beta)$,$(\alpha\wedge\beta)$,$(\neg\alpha)$都是命题。

(iii)一个符号串是一个命题,当且仅当它是从命题字母开始并重复应用(ii)的实例得到的。

接下来,定义一个*真值指派*:从原子命题 p 到真值$\mathcal{A}(p)$的映射,其中$\mathcal{A}(p)$取 1(表示真)或 0(表示假)(第一章的定义 3.1)。(这样一个指派对应于在填写一张真值表格的一行时,给命题字母 p 赋值为$\mathcal{A}(p)$。)每一个指派$\mathcal{A}$都有一个唯一扩张到一个*真值赋值*$\mathcal{V}$,它是从所有命题组成的集合到$\{0,1\}$的映射。根据上面定义中的归纳子句,将联结词替换为$\{0,1\}$上对应的*布尔运算*(Boolean operation),便可以确定这个赋值。这些运算通常由如下的表给定:

$$0\vee 0=0 \qquad 0\vee 1=1\vee 0=1\vee 1=1$$

$$1\wedge 1=1 \qquad 0\wedge 1=1\wedge 0=0\wedge 0=0$$

$$\neg 1=0 \qquad \neg 1=0。$$

如第一章定义 3.2 那样,对应于给定指派$\mathcal{A}$的赋值$\mathcal{V}$的归纳定义如下:

(i)$\mathcal{V}(p)=\mathcal{A}(p)$,如果 p 是一个命题字母。

(ii)对所有的命题 α, β,

$$\mathcal{V}(\alpha\vee\beta)=\mathcal{V}(\alpha)\vee\mathcal{V}(\beta),\quad \mathcal{V}(\alpha\wedge\beta)=\mathcal{V}(\alpha)\wedge\mathcal{V}(\beta),\quad \mathcal{V}(\neg\alpha)=\neg\mathcal{V}(\alpha)$$

(如第一章第二节所描述的那样,$\mathcal{V}$给出的值可由指派$\mathcal{A}$决定的真值表的那一行而得到。)因此一个命题可以看作是其命题字母值的所有取值的二值函数。这就是把联结词称为复合语句的真值泛函模式的用意。这是克利西普斯的想法的现代形式。在这种观点下,一个在任何真值赋值下都为 1 的命题称为*重言式*或*永真的*,也就是说,作为一个命题函数,它是取值为 1 的常值函数。这些是命题逻辑的逻辑真理。

在实数或复数域上通常的代数里,多项式代数和多项式函数的代数实际上是不可区分的(用现代的语言来说,就是同构)而且在布尔的时代,它们两个被看作是一样的。受到通常的多项式的影响,布尔自然把一个命题解释为其导出的二值命题函数。根据这种解释,逻辑运算"且"、"或"和"非"就与命题函数上的运算对应起来了。这样就把命题函数的集合变为一个代数结构,其中,如果两个命题对应于同一个命题函数,那么它们就是相同的命题。也就是说,两个命题 α 和 β,如果对任意一个真值赋值$\mathcal{V}$,都有$\mathcal{V}(\alpha)=\mathcal{V}(\beta)$(这意味着 α 和 β 有相同的真值赋值表),那么 α 和 β 是相同的,即 $\alpha=\beta$。例如,$\neg(\alpha\wedge\beta)=(\neg\alpha\vee\neg\beta)$。这个等式是布尔代数的定律之一。

布尔也发现了类代数和命题函数代数满足相同的定律。为了解释布尔关于命题和类演算关系的洞见,我们暂时采用 19 世纪弗雷格的一个自然的观点(下面我们将对之进行更加详尽的处理)。

对每一个性质 $P(x)$,我们都引入一个新的抽象对象 A,称之为类(class)。我们定义 $x\in A$,读作 x 是 A 的一个元素,来表示 $P(x)$为真。*概括公理*就是说每一个性质 P 决定一个类 A。*外延性公理*就是说它只决定唯一的一个类 A:如果 A 和 B 都是类,那么 $A=B$ 是指,对所有的 x,$x\in A$,当且仅当 $x\in B$。因此,对任意的性质 P,我们都可以指定一个由 P 决定的唯一的类,记作 $A=\{x\mid P(x)\}$。

布尔的类代数基于以下的运算:

1. 并：$A \cup B = \{x \mid x \in A \vee x \in B\}$

2. 交：$A \cap B = \{x \mid x \in A \wedge x \in B\}$

3. 补：$-A = \{x \mid x \notin A\}$

它们每一个都由概括公理保证其存在性而外延性公理决定其唯一性。然后类代数的定律就可以由命题函数的定律得到。例如，因为对每一个 x，由 $\neg P(x) \wedge Q(x))$ 和 $\neg P(x) \vee \neg Q(x)$ 决定的两个命题函数是一致的，而外延性公理蕴涵了 $-(A \cap B) = (-A) \cup (-B)$。

正是布尔发现了命题的真值表（第一章的第二节）和隐藏在“布尔展开定律”下的析取范式（第一章第二节习题和第四节习题 9）。正是他用纯代数给出了系统的命题逻辑推理，他的工作导致了逻辑代数（见参考文献第三节九）。正是布尔指出了全称和存在量词分别是最大下界和最小上界的例子，并且给它们引入了代数符号。这些概念被施罗德（Schröder，1841—1902）[1877，2.3]规范化并在 Schröder[1890—1905，2.3]中得到极大发展。但是在量词方面布尔并没有得到类似的好的理论。除了指出量词也是命题函数的运算外，他并没有改进亚里士多德对量词的处理。施罗德也发展了一个语句集合模型的概念。后来，勒文海姆（Löwenheim，1887—1940）[1915，2.3]继续研究他的工作。

第五节 19 世纪数学的基础

19 世纪也见证了用精确的定义、公理和构造来给数学建立一个牢固基础的共同努力。这一努力是由数学自身的困难所激发的。当定义不精确时，就会产生混淆和争论。在区分用符号表示的函数时困难重重。对于区分连续性和一致连续性、收敛性和一致收敛性、收敛性和可加性的形式、可微性和连续性等等，都存在困难。什么是一个整数？一个有理数？一个实数？一个函数？一个连续的函数？即使是早期分析严密性最伟大先驱之一的柯西，也因为“证明”了：无穷连续函数的级数和仍然是连续的，在 19 世纪 20 年代初期栽了一个跟头。1826 年，阿贝尔（Abel）给出了一个反例（关于这个证明精确的描述和它内在的错误，详见 Kitcher[1983，1.2]第 254 页）。那么数学的基础会是什么呢？

在欧几里得的《几何原本》中，把综合几何作为数学的逻辑基础，数学的每部分都归约为几何。在 17 世纪，笛卡儿把综合几何学归约到了解析几何学，从而通过引入坐标归约到了代数和数字。这导致了 19 世纪的大量工作都是用简单的数学来定义复杂的数学结构。在这期间我们发现如下定义：

1. 用非负整数对来定义整数；
2. 用整数对来定义有理数；
3. 用有理数集或序列来定义实数；
4. 把复数看作是实数对。

（这方面工作的重要贡献者有 1858 年魏尔斯特拉斯（Weierstrass，1815—1897）（见 DuGac[1973，1.2]），戴德金（Dedekind，1831—1916）1862—1863 年的微积分课程（见 DuGac[1976，1.2]）及 1872 年的文章[1963，2.3]、海涅（Heine，1821—1881）[1872，1.2]和 Cantor[1872，1.2]。要了解更多这方面的内容可见 Birkhoff[1973，1.1]和 Struik[1969，1.1]。

这一归约的过程被称为“魏尔斯特拉斯分析算术化”。最后，非负整数的公理化是戴德金（在 1872 年的文章[1963，2.3]）和皮亚诺（Peano，1858—1932）[1894—1908，2.3]与

[1990, 2.3]用一个包含元素0(或1)的集合和一个后继函数$S(x)$(加1)完成的。

为了形式化并且精确地来执行这个归约，有两个缺陷需要弥补。

1. 缺陷Ⅰ. 逻辑。由逻辑推理推出所有系统的所有性质。那么逻辑推理的公理和规则是什么呢?

2. 缺陷Ⅱ. 集合论。每一个系统都是从比较简单的系统用集合构造定义的，例如构造出两个集合的无序对，给定集合的所有子集的集合、一些集合的并集或者是某个给定的集合里具有某种给定性质的所有元素的集合。并且为了能继续下去我们必须在一开始就假定存在一些集合。需要假定哪些集合呢? 需要哪些集合构造法呢? 需要哪些集合论的公理呢?

这一情况可能被皮亚诺的工作概括了。综合19世纪用简单系统建立复杂系统的工作，皮亚诺(见[1973, 2.3])沿着上面描述的路线对当时的数学第一次进行了系统的整理。他给集合论构造和逻辑联结词与量词都引入了系统的符号，它们和今天所使用的几乎没什么不同。但在他的工作中却没有给出所需要的集合论构造公理和所使用逻辑的推理规则。仅仅假定它们是数学家所熟知的。值得注意的是，皮亚诺并没有想要把所有的东西一丝不苟地完全形式化。他想要创造出一种表示方法，使数学家们不用他们的母语就可以思考和交流，无论他们说英语、希腊语、法语或是德语。这方面他是很成功的。他甚至有更大的抱负。他还发展了一种通用的科学语言，并且曾经尝试将它发展成为一种通用的日常语言。这并不是很受欢迎。毕竟，改变少数数学家的习惯要比改变所有人的习惯更容易。

缺陷Ⅰ是被弗雷格填补的Frege[1879, 2.3]，他给出第一个包括量词、关系和命题联结词在内的谓词逻辑的形式化处理。结合亚里士多德对于量词的处理和布尔对于命题联结词和任意元的关系符号R的处理，他以形如$R(x_1, \cdots, x_n)$的原子公式为基础，应用“且”($\wedge$)、“或”($\vee$)、“非”($\neg$)等逻辑联结词和“对所有x_i”($\forall x_i$)及“存在一个x_i”($\exists x_i$)这样的量词，形式化了逻辑公式的概念。

现代符号中关于谓词逻辑语法的精确定义是在第二章第二节中给出的。下面给出它的一种简单定义，其中含有：命题逻辑的逻辑联结词、一组*变元*$x, y, z, x_1, y_1, \cdots$，与*量词*$\forall$和$\exists$以及对于每一个$n$，有一组关系符号$R, S, \cdots$(称之为*$n$-元关系符号*(*n*-ary relation symbol))：

(i)如果R是一个n-元关系符号并且$x_1, \cdots, x_n$是变元，那么$R(x_1, \cdots, x_n)$就是一个公式。这些称为*原子公式*。

(ii)如果φ和ψ是公式并且x是变元，那么$(\varphi \vee \psi)$，$(\varphi \wedge \psi)$，$(\neg\varphi)$，$(\forall x)\varphi$和$(\exists x)\varphi$也都是公式。

代数中的公式是一种用所考虑的特殊代数的基本或原子关系来表达复杂关系的方式。同样地，谓词逻辑中的公式直观地定义了出现在公式中的变元之间的“关系”。(事实上是自由变元之间的关系：公式φ中变元x的出现称为是*自由的*是指它不出现在形如$(\forall x)\varphi$或$(\exists x)\varphi$的φ的子公式中的。一个不出现任何自由变元的公式称为*语句*。)

Frege[1879, 2.3]也第一次给出了一些公理和推理规则，并且把证明定义为语句的一个有穷序列，其中每一个语句都是一个公理或是直接应用推理规则由前面的语句得到。在[1879, 2.3]中，他考虑在所有的对象中取值的量词。他还没有模型这个概念，在一个模型中给定了一个集合或者定义域并且变元在这个集合中取值。而现在的谓词逻辑语义的基础是由施罗德引入的，他是布尔在逻辑代数方面的继承者。

必须提到的是 C. S. Peirce[1870]也独立地发展了谓词逻辑的演绎结构，但它并不为人熟知。最后，我们要指出：亚里士多德的三段论已经被包含在了所谓的谓词逻辑的二量词片断(没有函数符号)中，它有一个逻辑永真性的判定方法，我们已经在正文的第二章第七节习题 10 中给出。

弗雷格同样也处理了缺陷Ⅱ。他给出了逻辑和数学的第一个完全形式化的基础，在 Frege[1903，2.3]中(也可以见[1953，2.3]和[1977，2.3])直接定义了正整数。他对于逻辑的处理是很完善的。我们可以认为类的处理是基于早期给出的一对直观形象的非逻辑公理，即把隶属 $\in$ 和恒等 $=$ 作为初始概念：

1. **外延性公理**：对所有的类 A，B，如果对所有的 x，$x \in A$，当且仅当 $x \in B$，那么$A = B$。

2. **概括公理**：对所有的性质 $P(x)$，存在一个类 A，对所有的 x，满足 $x \in A$，当且仅当$P(x)$。

根据外延性公理，概括公理中断言存在的类 A 是唯一的，我们记作 $A = \{x \mid P(x)\}$。(这两条公理和下面所讨论的公理及其在集合论中的有关发展，我们全都在第六章中以更长的篇幅进行了论述。)

弗雷格的愿望是任何一个能够在形式语言中被公式化的性质都被接受当作 P。类 A 同样也被考虑作为对象，能够作为任何一个类的元素。用这些公理，我们就能够马上推导出所有常用的数学对象和运算的存在性。例如，我们写出能够应用概括公理产生集合构造的定义，有经验的读者将能识别出它就是现代公理集合论的基础。在弗雷格的系统中，它们并没有用作独立的公理。

1. **空(零)类(Empty(Null)class)**：存在一个类$\varnothing = \{x \mid \neg(x = x)\}$。

2. **无序对**：给定两个类 A 和 B，存在一个类

$$\{A,B\} = \{x \mid x = A \text{ 或 } x = B\}$$

3. **并集公理**：给定一个类 A，存在一个类

$$\bigcup A = \{x \mid (\exists y)(y \in A \wedge x \in y)\}$$

4. **子集构造(分离)公理(Subset Construction(Aussonderung) Axiom)**：给定一个类 A 和任何一个性质 $P(x)$，存在一个类 $B = \{x \mid x \in A \wedge P(x)\}$。

5. **替换公理**：给定一个单值性质 $Q(x, y)$，即对每个 x，只存在一个 y，使得 $Q(x, y)$ 和任何类 A，存在一个类 $B = \{y \mid (\exists x)(x \in A \wedge Q(x, y))\}$。

6. **无穷公理**：存在一个最小归纳类(类 A 是*归纳的*(inductive)，如果$\varnothing \in A$ 并且对每个 x，$x \in A \rightarrow x \cup \{x\} \in A$。)注意，至少存在一个归纳类，即所有类的类$\{x \mid x = x\}$，并且归纳类的交仍然是归纳的。则最小归纳类可由概括公理给出：

$$\{x \mid (\forall y)(y \text{ 是归纳的} \rightarrow x \in y)\}$$

(ω 这个类是由冯·诺伊曼(1903—1957)在[1923，2.3]中作为断言存在整数集的公理而引入的。然而，这并不是弗雷格使用的整数的定义。冯·诺伊曼重新设计了弗雷格的原始定义，使得它能够自然推广到超穷序数上。如果弗雷格也想把超穷序数和基数包含在定义之内的话，冯·诺伊曼的定义大概也能被他接受吧。)

弗雷格的系统简单得令人着迷。直接写出你心中集合的描述。概括公理马上能保证它的存在性并且外延性公理确保它是唯一的。你可以看到弗雷格为什么会满意了。但是，很可

惜，当他的《算术原理》[1903, 2.3]的第2卷还在准备阶段时，罗素(1872—1970)的悖论出现了：

设 $P(x)$ 是性质 $\neg(x \in x)$，令 $A = \{x \mid P(x)\}$。应用 A 的这个定义到 A 自身可以得到：$A \in A$，当且仅当 $\neg(A \in A)$，一个显然的矛盾。

在弗雷格的名著完成的时候，他为数学基础建立的结构倒塌在了矛盾之中。没有限制的朴素的概括公理必须被摒弃。然而，上面列出的概括公理的实例和外延性公理一起，形成了现代集合论的基础(见参考文献列表3.3)。

19世纪90年代见证了康托关于基数和序数理论的伟大成就(Cantor[1952, 2.3])。罗素的悖论在这一成就上投下了暂时的阴影，因为它用了概括公理的实例，从而显示出来是基于类构造的。康托的理论具有内在的不相容性吗？康托自己已经在[1885, 2.3]关于弗雷格[1884, 2.3]的评论中把弗雷格没有限制的概括公理描述为基于不相容性质的。他表明他的理论仅仅是用了相容的性质来构造类。但是，他并没有指出如何区分相容与不相容的性质。这并不使他烦心，因为他认为他的工作和自然数一样都是基于数学直觉的，从而也有牢固的基础。他关心的并不是基于形式逻辑的公理集合论而是基于直觉和构造的数学。

第六节 20世纪数学的基础

根据20世纪初的有关数学基础发展的传统说法，罗素悖论和其他的悖论导致了三种不同的思想路线。一种被称为逻辑主义。这是怀特海(Whitehead, 1861—1947)和罗素[1910—1913, 2.3]试图解决弗雷格和康托的问题时创立的。他们通过把弗雷格的方法限制到类型理论上，将数学全面归约到逻辑。这在数学上是个难以使用的理论。

第二种思想路线被称为形式主义。它是由伟大的德国数学家希尔伯特(Hilbert, 1862—1943)在试图重建集合论的乐园时开创的，这个乐园是康托给出形式系统相容性的"有穷证明"时创建的，他所考虑的形式系统大到足够包括现代数学的各个分支如数论、分析学、集合论等。形式主义只是数学语法，已经远远脱离了这个术语原有的意思。数学的语句被简单的看作是字母串。公理都是串，推理规则变成了从串推导出串的规则，并且它变成了这些类似于象棋那样的规则是否会导致 $0 = 1$ 之类的矛盾就变成了一个组合问题。

第三种路线是直觉主义，其创立者是伟大的现代拓扑学的奠基者荷兰人布劳威尔(L. E. J. Brouwer, 1881—1966)(见Brouwer[1975, 2.3])。(我们在第五章讨论了直觉主义。)他认为数学事实的根源是直觉。这个康托和庞加莱也会同意，但布劳威尔还得出结论：证明定理时如果断言存在某物，只有给出了它的明确的构造，这个证明才是有意义的证明。对他而言，证明就是这些构造而定理仅仅表达了它们的结果。所以证明中的步骤就是构造的规则。他强调构造是数学的基本组成部分，从这个角度看他的观点是计算机科学的先驱，尽管这并不是他的意向。他不赞成像希尔伯特那样建立固定的形式系统。他认为新的构造规则可能在任何时候出现，而一个固定的系统却会把数学家限制在固定的一些构造之中。一般来讲，类似于康托超穷序数的工作，他认为受过教育的人凭直觉就可以立刻辨认出数学原理的正确性。

悖论对逻辑学和数学基础的影响，并没有像19世纪的数学所考虑的连续那样得到同样的重视。希尔伯特在他著作的[1899和1950, 2.3]《几何基础》中，第一个给出了欧氏几何的完备公理集。他彻底地检查了这些公理的相容性和独立性。书中提出了一种证明的框架：

即寻找足够多的公理来作为一个学科的基础，然后证明它们的相容性，他后来提议证明关于数论、分析学和集合论的相容性就是这个工作的一个自然推广。希尔伯特自己主要讨论越来越复杂的系统相容性的直接有穷证明。很多他的追随者则试图把像分析那样的复杂系统的相容性归约到像数论那样看似比较简单的系统中。在这里，他们效仿了用整数构造有理数、再用有理数构造实数的做法。这个计划因为哥德尔的不完全性定理[1931，2.3]而宣告失败。

那个时候，证明相容性的典型模式是用欧氏几何解释非欧几何的公理，把非欧几何的相容性归约到欧氏几何的相容性。通过把欧氏几何解释成分析，希尔伯特自己给出了欧氏几何相容性的证明。然后他希望通过把分析解释到有穷算术系统中来证明分析的相容性，这样它的相容性在有穷系统中是可证的。而有穷系统自身的相容性能够由直觉得到。

对布劳威尔来说，他指出他对构造性方法的偏好是来自于年轻时的爱好，而不是因为对悖论产生的怀疑。他的前辈19世纪代数学家克罗内克(Kronecker，1823—1891)就是一个极好的例子，因为克罗内克坚持认为对于代数几何和数论中的一切给出明确的算法。克罗内克最喜欢的格言是“上帝创造了整数，其他一切都是人造的”。克罗内克的意思是每一个数学对象都可以用整数来构造，而整数是由直觉直接给出的。克罗内克对当时的代数进行了构造性的处理，而布劳威尔把他的方法推广到了分析和拓扑。1967年毕晓普(E. Bishop，1928—1983)(见 Bishop 和 Bridges[1985，4.2])把这个构造性处理推广到了我们所知道的现代分析学。

由康托发起的集合论数学，在当时仍然有其他强有力发展趋势。策梅洛(Zermelo，1871—1953)[1904，2.3]发表了一个关于每个集合都可以被良序化的著名证明：也就是说每个非空集合都可以被排序，从而每个非空子集都有最小元。后者是康托没有证明而直接使用的一个原理。策梅洛的证明因为两个原因饱受批评。首先，策梅洛(非常有意识地)使用了一条本质上是新的集合论公理，即现在所谓的选择公理：

选择公理：任意给定一些互不相交的非空集合，存在另一个集合，它和每一个集合都恰好有一个公共元素。

其次，他用了当时还没有严格定义的康托序数。后来他又在不假定康托序数的条件下给出了第二个证明。然后，他又重新回到了集合论以展示出其公理，其中之一显然是先前所遗漏的。这就是策梅洛的集合论[1908，2.3]。策梅洛集合论是在没有潜在逻辑的形式化的情况下，非形式的集合构造公理的一个集。这些公理对于构造大序数和大基数是不够的。我们前面非形式地讨论过的替换公理弥补了这一缺陷，这个工作是由弗兰克尔(Fraenkel，1989—1965)[1922，2.3]和斯科朗(Skolem，1887—1963)[1922，2.3]完成的。序数、基数以及超穷归纳的一个令人满意的处理最后是由冯·诺伊曼(van Neumann，1903—1957)[1923，2.3]和[1925，2.3]补充到 Zermelo-Fraenkel 集合论中的(见 van Heijenoort[1967，2.1])。

最后，策梅洛-弗兰克尔-斯科朗集合论成了20世纪数学基础之争的胜利者。我们注意到当最后一个数学公理依序出现，即替换公理、逻辑的形式应用和集合论的形式表示走到了一起，这是两者在得到数学基础的精确性中都是必须的一个指示。这个公理是说：如果 $\varphi(x, y)$ 是集合论中的一个(带参数的)公式，使得对任何 x，至多存在一个 y，有 $\varphi(x, y)$，那么对任意集合 A，存在一个由所有满足下面条件的 y 构成的集合 B：y 与 A 中的某个元素 x 满足 $\varphi(x, y)$。换句话说，这个公理说的就是在一个由公式定义的单值关系之下，集合的像

仍然是一个集合。这里的逻辑公式 $\varphi(x, y)$是必不可少的。

第七节 20世纪早期的逻辑

现在回到纯逻辑。Post[1921, 2.3]参照怀特海和罗素的《数学原理》[1910—1913, 2.3]给出了命题逻辑形式可证性的一个定义并且通过调整他的证明过程证明了一些基本的定理：

1. 可靠性(第一章定理5.1)：在所有的真值赋值中一个有证明的命题总是真的。
2. 完全性(第一章定理5.3)：在所有真值赋值中都真的命题有一个证明。

谓词演算发展到同样的地步经历了较长的时间。

在19世纪90年代，施罗德已经澄清了谓词逻辑中解释或结构的思想。一个结构包含了一个非空集合(即定义域)、一个集合上的关系(对应于语言中每个关系符号)和一个集合上的函数(对应于语言中每个函数符号)。量词的范围是特定的域。(见第二章第四节)在一个结构中，很自然地，问哪些语句为真是有意义的。一个逻辑上永真的语句是在所有关系符号的所有解释的所有定义域中都为真，也就是说对于语言所有的结构中都为真。

Löwenheim[1915, 2.3]用施罗德的想法和符号证明了：对谓词逻辑中的任何语句 S，如果 S 在某个定义域为 D 的结构中为真，则存在一个 D 的可数子集 D'，使得当一个结构是由 D 上的关系限制到 D'上形成的，S 在 D'中也为真。这个结果说明了不可数的实数是不能由谓词逻辑中一个简单句来刻画的。Skolem[1922, 2.3]简化了这个证明并把定理从一个语句推广到了可数多个语句(见第二章的定理7.7)。他发现：因为用谓词逻辑表达的集合论是可数的语句集合，即使集合论证明了存在不可数的集合，还是存在一个有可数定义域的集合论模型(也就是语言的一个结构，这个语言中所有公理都为真)。人们称之为斯科朗悖论，尽管它不是一个真正的悖论(见 Skolem[1922, 2.3]和 Kleene[1952, 3.2]，关于"悖论"的一个特别好的讨论见 Fraenkel 和 Bar-Hillel[1958, 3.3])。

斯科朗注意到每一个语句都可满足等价于一个所谓的前束语句，即带有所有的量词的语句在一个无量词公式的前面(见第二章第九节)。(可满足等价意味着一个语句有一个模型当且仅当另一个也有。他和勒文海姆都没有给出证明规则。)为了弄清楚他们做了什么工作，让我们来看一个形如 $(\forall x)(\exists y)\varphi(x, y)$ 的前束语句 S。假定 S 在定义域 D 中为真。因为 S 在 D 中为真，所以对每个 $x \in D$，可选择一个 y 满足 $\varphi(x, y)$。我们称这个元素为 $y(x)$，从而就定义了一个从 D 到 D 的函数 y。令 x_0 是 D 中任意一个元素。我们所要的子集 D'现在可以取作为 $\{x_0, y(x_0), y(y(x_0)), \cdots\}$。这个做法强调了，函数符号和定义域中的个体元素的名字是有用的。所以，在我们的谓词逻辑语言中包括了一组原始 n-元函数符号，$n=1, 2, 3, \cdots$，一个个体常元符号(individual constant symbol)的集合，这些符号是 D 中元素预期的名字。然后如第二章第二节那样定义项的集合：

(i)所有变元和常元符号都是项。

(ii)如果 f 是一个 n-元函数符号，并且 $t_1, \cdots, t_n$ 都是项，那么 $f(t_1, \cdots, t_n)$仍然是一个项。

然后，我们把谓词逻辑公式类按如下方式推广，即对任何 n-元关系符号 R 和任何项 $t_1, \cdots, t_n$，都把 $R(t_1, \cdots, t_n)$看作是一个原子公式。

实际上，勒文海姆-斯科朗式(如上)的构造把项的集合变成了结构。

通常认为，Gödel[1930，2.3]首先证明了谓词逻辑的完全性(第二章定理 7.8)：每一个在所有结构中为真的谓词逻辑语句都有一个谓词逻辑证明。Henkin[1949，3.4]对这个定理的证明(也可见 Chang 和 Keisler[1990，3.4])现在是模型论的一个基本事实，证明中直接构造了一个相容理论的模型。这一课题起源于斯科朗和勒文海姆的早期工作，但是它的真正发展却是源于塔斯基(Tarski)及其学派后来的工作(见参考文献第三节四)。

厄布朗(Herbrand，1908—1931)[1930，2.3]中包含了完全性定理证明的所有要素。他把每个谓词逻辑语句$\neg\varphi$都联系到命题逻辑语句的一个无穷序列ψ_n，并且指出$\neg\varphi$是可证的，当且仅当存在一个n，使得$\psi_1 \vee\cdots\vee \psi_n$是一个重言式。因此，如果$\neg\varphi$是不可证的，那么对每个$n$存在一个命题逻辑赋值，使得$\neg\psi_1 \wedge\cdots\wedge \neg\psi_n$为真。他也说明了怎样从所有的这些合取式的一个模型得到$\neg\varphi$的一个模型，从而证明了完全性定理。厄布朗不愿意用非构造性的方法，从每个有穷合取的真值赋值生成能够满足这些合取式的单个结构和赋值，从而使φ为假。厄布朗知道这么做会有多么的“容易”。然而，他把这些概念和做法视为不能容许的元数学。

让我们用自己的观点以一个特殊例子来简要地描述一下厄布朗的构造方法：$\neg\forall x\exists y\varphi(x, y)$的不可证性(从而有$\forall x\exists y\varphi(x, y)$的可满足性)。(第二章第十节有更多的细节和某种意义上更一般的情形。)厄布朗使用了基本项(没有变元的项)来构造所需要的命题序列。我们也用它们作为所需模型的定义域。这些项的集合现在被称为*厄布朗域*。

在斯科朗-勒文海姆定理的讨论中，我们接着使用上面引入的符号。引入一个常元c来表示D的元素x_0，并用函数符号f来表示扩充语言中的$y(x)$。那么，上面描述的模型也可以表示为由基本项的集合$\{c, f(c), f(f(c)), \cdots\}$构成的厄布朗域。在$\varphi$中出现的$n$-元关系符号$R$就可以表示所有的基本项$n$-元组$(t_1, \cdots, t_n)$的集合，使得$R(t_1, \cdots, t_n)$在上面描述的结构$D$中为真。因此，视每一个原子句$R(t_1, \cdots, t_n)$为一个命题字母，我们就得到命题逻辑的一个真值赋值，满足$R(t_1, \cdots, t_n)$在初始结构中为真就指定真值，否则为假。

这暗示着我们(从建立一个$\forall x\exists y\varphi(x, y)$的模型)着手的谓词逻辑问题可以由一个命题逻辑问题代替，它是关于厄布朗域中原子命题的真值赋值的。也就是说，存在一个使得$(\forall x)(\exists y)\varphi(x, y)$为真的结构，当且仅当存在一个如下没有量词的语句的(无穷)集合的命题真值赋值：

$$\begin{array}{ll} \psi_1: & \varphi(c, f(c)) \\ \psi_2: & \varphi(f(c), f(f(c))) \\ \psi_3: & \varphi(f(f(c)), f(f(f(c)))) \\ & \cdots\cdots \end{array}$$

使得所有这些命题逻辑语句同时为真。命题逻辑的紧致性定理(第一章的定理 6.13)说：存在一个真值赋值$\mathcal{V}$，使得命题逻辑语句的一个无穷序列$\{\psi_n\}$同时为真，当且仅当对每个n，存在一个真赋值$\mathcal{V}_n$，使得$\psi_1, \cdots, \psi_n$同时为真。根据这个定理，存在一个带$(\forall x)(\exists y)\varphi(x, y)$的结构，或等价地$(\forall x)\varphi(x, f(x))$为真，当且仅当对所有的$n$，存在一个命题逻辑赋值，使得$\psi_1 \wedge\cdots\wedge \psi_n$为真。这是厄布朗定理的语义版本。

厄布朗不愿意使用非构造性的方法，因此没有证明完全性定理：因为$\neg\varphi$的不可证明性蕴涵了存在φ的一个模型，所以存在一个证明程序，它产生的所有语句在任何结构中都是

永真的。然而，正是他对构造性方法的坚持，才使得他的工作那么重要。正是厄布朗而不是哥德尔的方法，特别是他的把谓词逻辑中的可证明性归约到命题逻辑，激发了经典逻辑自动化的研究并为之奠定了最初的基础；这个自动化促进了 PROLOG、专家系统和智能数据库等方面的发展。

第八节　推理和计算

希尔伯特是形式化公理方法的倡导者。他第一个给出了欧几里得几何学公理的完备集并且促进了一个很多数学分支的抽象观点。在 20 世纪 20 年代，他对于数学的逻辑就采取这样的观点。他开始研究形式系统，每一个都由一些公理及推理规则所定义，并开创了一门新的数学分支：元数学(见克林(Kleene，1909—1994)[1952，3.2])。特别地，他提出了一个计划：给诸如怀特海和罗素的形式系统或者策梅洛-弗兰克尔集合论的相容性找到一个数学证明，也就是说不可能推导出 $0=1$。

希尔伯特希望以这样的方式来分析这些系统中的证明，即能够给出一个有穷证明(即足够基本从而能够被普遍地接受)而不引起任何矛盾。他那个学派后来的追随者们的一种典型的研究方法是：对每个形式系统，找到一个性质 P，满足

(i) P 满足所有的公理。

(ii) 只要 P 满足某个推理规则的前提条件，它就满足相应的结论。

(iii) P 不满足 $0=1$。

他们期望在形式系统中通过对证明长度的归纳，使这样的证明能够进行下去。而且，(i) ~ (iii) 的证明必须是“有穷的”。

在希尔伯特和阿克曼(Ackermann)合著的逻辑学教科书中(Hilbert 和 Ackermann[1928，2.3])，选择了谓词逻辑作为研究对象并且在相容性的证明中明确强调了这一观点。他的学生和追随者们给出了一些简单的形式系统相容性的证明。这个计划导致了一门新的现代学科：证明论(见参考文献第 3 节五)。

希尔伯特计划的目标是令人信服地证明我们所知道的形式数学是相容的，从而把我们从像罗素悖论之类的困境中解救出来。这个目标在哥德尔发现他的不完全性定理[1931，2.3]之后很大程度上被放弃了。在不完全性定理的证明之后，希尔伯特仍然认为对他的有穷证明的概念进行加工可以解决其存在的问题(见 Hilbert 和 Bernays[1934 和 1939，2.3])。然而，有一种形式的哥德尔定理表明，对于足够“丰富”的数学理论，比如集合论或数论，除非它们自己是不相容的，否则它们不包含能证明自身相容性的方法。因为这些系统通常能使我们形式化所有已知可信的基本论证，我们不可能给这样丰富的系统找到其相容性的证明，除非使用确实更加强大的系统。这已经成为最近 60 年证明论的趋势。

如今依照惯例，假设一个理论是“足够丰富”的，是说这个理论的演绎能力强大到可以模拟任何算法的计算步骤。正是“算法可计算”这一数学定义的形式化，标志了现在称为递归论或可计算性理论这一逻辑学分支的建立。我们必须提到哥德尔、厄布朗、丘奇(Church，1903—1995)、图灵(Turing，1912—1954)、克林、波斯特(1897—1954)和马可夫(Markov，1903—1979)的名字，他们是这个学科的奠基人。早期重要的文章收集在 Davis[1965，2.1]中。递归函数(即那些用算法能行可计算的函数)的普遍被接受的形式定义第一次提出那些寻求算法的经典问题可能存在否定的答案。第一个这样的问题来自于可计算性理论自身，即

停机问题：不存在一种算法判定一台给定计算机器对于给定输入的计算是否会停止(第三章的定理 8.8)。在这第一个例子之后，很多来自于数学和计算机等各种领域的问题被证明是没有算法可解的。丢番图方程也许是最古老的一个，它在希尔伯特 1900 年列出的那些基本问题中以希尔伯特第十问题而闻名：找到一种算法来判定一个整系数的多项式方程(有多个变元)是否有整数解(见 Browder[1976，1.2]，第 1 ~ 34 页)。建立在普特南(Putnam，1931—)、戴维斯(Davis，1928—)和鲁宾逊(Julia Robinson，1919—1985)的基本工作之上，马蒂亚塞维奇(Matijacevič，1949—)最后完成了关于这个问题不可解的证明的最后步骤：不存在这样的算法(见 Davis，Matijacevič和 Robinson[1976，3.6])。这类证明典型地把所考虑的问题(如丢番图方程的解)归约到了停机问题的不可解性上。确实，从我们现在的观点来看，用演绎模拟某个算法是谓词逻辑中不完全性和不可判定性定理(第三章的推论 8.10)证明的核心。

我们来考虑这么一个例子：在一阶算术中通过演绎模拟算法，在哥德尔看来是原始递归函数的可表示性[1931，2.3]，而 Kleene[1936，3.6]推广到所有的递归函数。这个版本的算术是从下列符号开始的谓词逻辑理论：常元符号 0、后继函数 s、二元函数加法 + 和乘法 · 以及二元关系相等 =。它包括 + 和 · 通常的归纳定义：

$$0 + x = x \quad x + s(y) = s(x + y)$$

$$x \cdot 0 = 0 \quad x \cdot s(y) = x \cdot y + x$$

还有一个公理：$s(x) = s(y)$ 蕴涵了 $x = y$，通常的等词公理(见第三章第五节)和归纳公理，即对每个公式 $\varphi(x) = \varphi(x, y_1, \cdots, y_n)$，断言

$$((\varphi(0) \wedge ((\forall x)\varphi(x) \rightarrow \varphi(s(x)))) \rightarrow (\forall x)\varphi(x))$$

设 n 是对应于整数 n 的(数字)项。即 1 是 $s(0)$，2 是 $s(s(0))$ 等等。哥德尔和克林证明的是：对任何递归函数 f，存在这种语言的一个公式 $\varphi(x, y)$，满足 $f(m) = n$，当且仅当存在从这个理论的公理开始的 $\varphi(m, n)$ 的一个证明。这个定理是说一阶算术的演绎设备，即用软件实现的演绎机，可以通过推理 $\varphi(m, n)$ 来计算 f。在第三章的第八节中对不可判定性进行了类似的研究。这个并不是高效的计算方式，但它确实表明把演绎等同于计算在 20 世纪的逻辑学里并不是一个新鲜的主题。在计算机科学中，大量当代的逻辑处理各种这样的情形：其中一个演绎就是一个程序而一个程序也就是一个演绎，在直觉主义逻辑中尤其如此，并把这种原理称为 Curry-Howard 同构(见 Girard[1989，3.5])。

在过去的几十年中，递归论远远走出了这样简单的问题，即直接算法可判定性，来发展计算的通用理论、能行可定义性和计算的相对复杂性的一整套理论。我们建议读者阅读参考文献第三节六列出的一些基础教材。

第九节　逻辑自动化和 PROLOG 的现状

自从数字化计算机在 20 世纪 50 年代晚期问世以来，戴维斯和普特南[1960，5.7]以及很多其他研究者试图实现谓词逻辑中的自动化定理证明。(自 1960 年以来，所有有关的文章都收录在 Siekmann 和 Wrightson[1983，2.1]中。)他们的结果被他们的证明方法和计算机的性能(还比不上现在最小的个人电脑的功能)所限制。在他们工作的启发下，J. A. Robinson(1930—)于 1963 年给出了厄布朗定理的一个不同的实现方法(同样见 Siekmann 和 Wrightson[1983，2.1])。在[1965，5.7]中，鲁宾逊引入了消解方法(第一章第八节)作为命题演算的

独立的逻辑证明方法。其中包括一个最初出现在 Herbrand[1930, 2.3]中的算法，现在称为合一算法(第二章算法 12.3)。在考虑谓词逻辑的机器证明的时候，鲁宾逊的方法削减了相当数量的情形，用马丁·戴维斯(Martin Davis)的话来说就是“客观地说是改革了这一学科”(Davis[1983, 5.7]，第 19 页)。

20 世纪 60 年代的这些早期的工作产生了一些有趣的可机器实现的演绎方法(第二章第十三节)，但是由于它们的速度太慢，除了用于研究之外，没有什么用处。在 20 世纪 70 年代早期，Kowalski[1974, 5.4]建立于考尔麦劳厄(Colmerauer)和其他人[1974, 5.4]关于规划系统方面的工作，提出了仅对叫做“Horn 子句”的特殊公式的演绎进行机械化的想法。Horn 子句(第一章的第十节和第二章的第五节)都形如 $R_1 \wedge \cdots \wedge R_n \to T$，其中 $R_1, \cdots, R_n$ 和 T 都是原子语句。众所周知，Horn 子句的演绎可以计算所有的递归函数。这是 Kreisel 和 Tait[1961, 3.6]和 Smullyan[1961, 3.6]的基础。然而，即使是谓词逻辑的这么一个小片段在实际操作中仍然太慢。现在的发展结果是 PROLOG，一种计算机语言，它把 Horn 子句的证明方法限制在一类很特殊的演绎，即 SLD-树上(第三章第一节和第二节)。尽管这一演绎方法的标准实现足以计算所有的递归函数(第三章推论 8.7)，但它们仍然是逻辑不完全的(见第二章第二节)。

第十节　对未来的展望

自从微积分的代数和符号算法发展起来以后，它已经成为了一种普遍使用的工具。随着高性能计算的出现，它现在要解决的问题在规模和难度上都是一百年前难以想象的。在实践和应用越来越多地使用高性能计算的同时，我们看到自动化推理将来有着类似的发展前景。现在已经出现了诸如 Constable 的 NUPRL[1986, 5.6]、Huet 和 Coquand 的 CONSTRUCTIONS(见 Huet and Plotkin[1991, 5.7])等“数学助手”，它们能够明白引理、证明、战术和策略。我们希望，自动化推理可以像微积分那样，有着许多不同的应用。这不过是莱布尼茨梦想的一个现代推广。

进一步阅读建议

关于数学史，Stillwell[1989, 1.2]，Boyer[1989, 1.2]和 Edwards[1979, 1.2]都有很好的介绍。Kitcher[1983, 1.2]是一本非常有趣的关于数学本质的哲学论述，(第十章)还包含了一个简短的关于分析学发展的研究，作为当时哲学力量的一个实例。关于现代数学的历史和哲学方面的各种话题，有一本很好的合集是 Aspray 和 Kitcher[1988, 1.2]。

对于早期的逻辑而言，Kneale 和 Kneale[1975, 2.1]是很好的参考资料。对 20 世纪的逻辑，首先要参考 van Heijenoort[1967, 2.1]。在没有更多背景的情况下，读一读 1900 年以前的参考书如布尔、亚里士多德、莱布尼茨或罗素的著作会有益处。直到科恩(Cohen)的工作以前关于集合论历史的一本精彩而范围又相当广泛的评论是 Kanamori[1996, 2.2]。

这里是逻辑方面一些基本的现代论文。它们是由 van Heijenoort[1967, 2.1]中那些一流的专家翻译和介绍的。在读完前两章或者浏览了本章之后，这些文章就能够为学生们所阅读了：

Fraenkel, A. A., “The notion ‘definite’ and the independence of the axiom of choice”[1922].

Frege, G., "Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought" [1879].

Gödel, K., "The completeness of the axioms of the functional calculus of logic" [1930].

Gödel, K., "On formally undecidable propositions of Principia Mathematica" [1931].

Herbrand, J., *Investigations in Proof Theory*, University of Paris [1930].

Löwenheim, L., "On possibilities in the calculus of relatives" [1915].

Peano, G., "The principles of arithmetic, presented by a new method" [1889].

Skolem, T., "Logico-Combinatorial investigations in the satisfiability or provability of mathematical propositions" [1920].

von Neumann, J., "An axiomatization of set theory" [1925].

Post, E., L., "Introduction to a general theory of elementary propositions" [1921].

Zermelo, E., "Proof, that every set can be well-ordered" [1904].

Zermelo, E., "Investigations in the foundations of set theory 1" [1908].

这里是一些收集在 Siekmann 和 Wrightson[1983, 2.1]中关于逻辑自动化的文章。同样在读完前两章或者浏览了本章之后，这些文章就能够为学生们所阅读了：

Davis, M., "The prehistory and early history of automated deduction" [1983].

Davis, M., "A computer program for Pressburger's algorithm" [1957].

Davis, M., and Putnam, H., "A computing procedure for quantification theory" [1960].

Davis, M., "Eliminating the irrelevant from mechanical proofs" [1963].

Gelernter, H., "Realization of a geometry-theorem proving machine" [1959].

Gilmore, P. C., "A proof method for quantification theory: its justification and realization" [1960].

Robinson, J. A., "Theorem-proving on the computer" [1963].

Robinson, J. A., "A machine-oriented logic based on the resolution principle" [1965].

Wang, H., "Proving theorems by pattern recognition I" [1960].

Wos, L., Robinson, G. A. and Carson, D. F., "Efficiency and completeness of the set of support strategy in theorem proving" [1965].

另外一个相关文章的很好来源是 Bledsoe 和 Loveland[1984, 5.1]。

附录 B　一个家谱数据库

下面给出形如“fatherof(a, b)”的列表是从旧约-历代记(希伯来圣经的最后一本)的前面几章中摘出的家谱信息。我们在许多程序设计问题和项目中用到它们，包括第二章的第五节习题 7 ~ 8 和第三章的第二节习题 12 ~ 14。提醒：当在使用这些数据运行语义校正的程序时，会出现许多不可预料的结果。第三章的第二节习题 14 列出了其中一些可能发生的问题的根源。这些问题往往是现实生活中所特有的。解决这些问题是有趣而且有意义的练习。

```
fatherof(adam,seth).
fatherof(seth,enosh).
fatherof(enosh,kenan).
fatherof(kenan,mahalalel).
fatherof(mahalalel,jared).
fatherof(jared,enoch).
fatherof(enoch,methuselah).
fatherof(methuselah,lamech).
fatherof(lamech,noah).
fatherof(noah,shem).
fatherof(noah,ham).
fatherof(noah,japheth).
fatherof(japheth,gomer).
fatherof(japheth,magog).
fatherof(japheth,madai).
fatherof(japheth,javan).
fatherof(japheth,tubal).
fatherof(japheth,meshech).
fatherof(japheth,tiras).
fatherof(gomer,ashkenaz).
fatherof(gomer,diphath).
fatherof(gomer,togarmah).
fatherof(javan,elishah).
fatherof(javan,tarshish).
fatherof(javan,kittim).
fatherof(shelah,eber).
fatherof(eber,peleg).
fatherof(eber,joktan).
fatherof(joktan,almodad).
fatherof(joktan,shelelph).
fatherof(joktan,hazarmaveth).
fatherof(joktan,jerah).
fatherof(joktan,hadoram).
fatherof(joktan,uzal).
fatherof(joktan,diklah).
fatherof(joktan,ebal).
fatherof(joktan,abimael).
fatherof(joktan,sheba).
fatherof(joktan,ophir).
fatherof(joktan,havilah).
fatherof(joktan,jobab).
fatherof(javan,rodanim).
fatherof(ham,cush).
fatherof(ham,mizraim).
fatherof(ham,put).
fatherof(ham,canaan).
fatherof(cush,seba).
fatherof(cush,havilah).
fatherof(cush,sabta).
fatherof(cush,raama).
fatherof(cush,sabteca).
fatherof(raamah,sheba).
fatherof(raamah,dedan).
fatherof(cush,nimrod).
fatherof(canaan,sidon).
fatherof(canaan,heth).
fatherof(shem,elam).
fatherof(shem,asshur).
fatherof(shem,arpachshad).
fatherof(shem,lud).
fatherof(shem,aram).
fatherof(shem,uz).
fatherof(shem,hul).
fatherof(shem,gether).
fatherof(shem,meshech).
fatherof(arpachshad,shelah).
fatherof(midian,ephah).
fatherof(midian,epher).
fatherof(midian,enoch).
fatherof(midian,abida).
fatherof(midian,eldaah).
fatherof(abraham,isaac).
fatherof(isaac,esau).
fatherof(isaac,israel).
fatherof(esau,eliphaz).
fatherof(esau,reuel).
fatherof(esau,jeush).
fatherof(esau,jalam).
fatherof(esau,korah).
fatherof(eliphaz,teman).
fatherof(eliphaz,omar).
fatherof(eliphaz,zephi).
```

```
fatherof(shem,arpachshad).
fatherof(arpachshad,shelah).
fatherof(shelah,eber).
fatherof(eber,peleg).
fatherof(peleg,reu).
fatherof(reu,serug).
fatherof(serug,nahor).
fatherof(nahor,terah).
fatherof(terah,abraham).
fatherof(abraham,isaac).
fatherof(abraham,ishmael).
fatherof(ishmael,nebaioth).
fatherof(ishmael,kedar).
fatherof(ishmael,abdeel).
fatherof(ishmael,mibsam).
fatherof(ishmael,mishma).
fatherof(ishmael,dumah).
fatherof(ishmael,massa).
fatherof(ishmael,hadad).
fatherof(ishmael,tema).
fatherof(ishmael,jetur).
fatherof(ishmael,naphish).
fatherof(ishmael,kedmah).
fatherof(abraham,zimran).
fatherof(abraham,jokshan).
fatherof(abraham,medan).
fatherof(abraham,midian).
fatherof(abraham,ishbak).
fatherof(abraham,shuah).
fatherof(jokshan,sheba).
fatherof(jokshan,dedan).
fatherof( ezer, jaakan ).
fatherof(dishan,uz).
fatherof(dishan,aran).
fatherof(israel,reuben).
fatherof(israel,simeon).
fatherof(israel,levi).
fatherof(israel,judah).
fatherof(israel,isachar).
fatherof(israel,zebulun).
fatherof(israel,dan).
fatherof(israel,joseph).
fatherof(israel,benjamin).
fatherof(israel,naphtali).
fatherof(israel,gad).
fatherof(israel,asher).
fatherof(judah,er).
fatherof(judah,onan).
fatherof(judah,shelah).
fatherof(judah,perez).
fatherof(judah,zerah).
fatherof(perez,hezron).
fatherof(perez,hamul).
fatherof(zerah,nri).
fatherof(eliphaz,gatam).
fatherof(eliphaz,kenaz).
fatherof(eliphaz,timna).
fatherof(eliphaz,amalek).
fatherof(reuel,nahath).
fatherof(reuel,zerah).
fatherof(reuel,shammah).
fatherof(reuel,mizzah).
fatherof(seir,lotan).
fatherof(seir,shobal).
fatherof(seir,zibeon).
fatherof(seir,anah).
fatherof(seir,dishon).
fatherof(sier,ezer).
fatherof(seir,dishan).
fatherof(lotan,hori).
fatherof(lotan,homam).
fatherof(shobal,alian).
fatherof(shobal,manahath).
fatherof(shobal,ebal).
fatherof(shobal,shephi).
fatherof(shobal,onam).
fatherof(zibeon,aiah).
fatherof(zibeon,anah).
fatherof(anah,dishon).
fatherof(dishon,hamran).
fatherof(dishon,eshban).
fatherof(dishon,ithran).
fatherof(dishon,chran).
fatherof(ezer,bilhan).
fatherof(ezer,zaavan).
fatherof(caleb, jesher).
fatherof(caleb,shobab).
fatherof(caleb,ardon).
fatherof(caleb,hur).
fatherof(hur,uri).
fatherof(uri,bezalel).
fatherof(hezon,segub).
fatherof(segub,jair).
fatherof(machir,gilead).
fatherof(hezron,ashhur).
fatherof(ashhur,tekoa).
fatherof(jerahmeel,ram).
fatherof(jerahmeel,bunah).
fatherof(jerahmeel,oren).
fatherof(jerahmeel,ozem).
fatherof(jerahmeel,ahijah).
fatherof(jerahmeel,onam).
fatherof(ram,maaz).
fatherof(ram,jamin).
fatherof(ram,eker).
fatherof(onam,shammai).
fatherof(onam,jada).
fatherof(shammai,nadab).
```

```
fatherof(zerah,ethan).
fatherof(zerah,heman).
fatherof(zerah,calcol).
fatherof(zerah,dara).
fatherof(nni,achar).
fatherof(ethan,azariah).
fatherof(hezron,jerahmeel).
fatherof(hezron,ram).
fatherof(hezron,chelubai).
fatherof(ram,amminadab).
fatherof(anninadab,nahshon).
fatherof(nahshon,salma).
fatherof(salma,boaz).
fatherof(boaz,obed).
fatherof(obed,jesse).
fatherof(jesse,eliab).
fatherof(jesse,abinadab).
fatherof(jesse,shimea).
fatherof(jesse,nethanel).
fatherof(jesse,raddai).
fatherof(jesse,ozem).
fatherof(jesse,david).
fatherof(jether,amasa).
fatherof(hezon,caleb).
fatherof(jekamiah,elishama).
fatherof(caleb,meshah).
fatherof(mareshah,hebron).
fatherof(meshah,ziph).
fatherof(hebron,korah).
fatherof(hebron,tappuah).
fatherof(hebron,rekem).
fatherof(hebron,shema).
fatherof(shema,raham).
fatherof(raham,jorkeam).
fatherof(rekem,shammai).
fatherof(shammai,maon).
fatherof(maon,bethzur).
fatherof(caleb,haran).
fatherof(caleb,moza).
fatherof(caleb,gazez).
fatherof(haran,gazez).
fatherof(jahdai,regem).
fatherof(jahdai,geshan).
fatherof(jahdai,pelet).
fatherof(jahdai,ephah).
fatherof(jahdai,shaaph).
fatherof(caleb,sheber).
fatherof(caleb,tirhanah).
fatherof(caleb,shaaph).
fatherof(shaaph,madmannah).
fatherof(shammai,abishur).
fatherof(abishur,ahban).
fatherof(abishur,molid).
fatherof(nadab,seled).
fatherof(nadab,appaim).
fatherof(appaim,ishi).
fatherof(ishi,sheshan).
fatherof(sheshani,ahlai).
fatherof(jada,jether).
fatherof(jada,jonathan).
fatherof(jonathan,peleth).
fatherof(jonathan,zaza).
fatherof(jarha,attai).
fatherof(attai,nathan).
fatherof(nathan,zabad).
fatherof(zabad,ephlal).
fatherof(ephlal,obed).
fatherof(obed,jehu).
fatherof(jehu,azariah).
fatherof(azariah,helez).
fatherof(helez,eleasah).
fatherof(eleasah,sisamai).
fatherof(sisamai,shallum).
fatherof(shallum,jekamiah).
fatherof(david,elishama).
fatherof(david,eliphelet).
fatherof(david,nogah).
fatherof(david,nepheg).
fatherof(david,japhia).
fatherof(david,elishama).
fatherof(david,eliada).
fatherof(david,eliphelet).
fatherof(solomon,rehoboam).
fatherof(rehoboam,abijah).
fatherof(abijah,asa).
fatherof(asa,jehoshaphat).
fatherof(jehoshaphat,joram).
fatherof(joram,ahaziah).
fatherof(ahaziah,joash).
fatherof(joash,amaziah).
fatherof(amaziah,azariah).
fatherof(arariah,jotham).
fatherof(jotham,ahaz).
fatherof(ahaz,hezekiah).
fatherof(hezekiah,manasseh).
fatherof(manasseh,amon).
fatherof(amon,josiah).
fatherof(josiah,johanan).
fatherof(josiah,jehoiakim).
fatherof(josiah,zedekiah).
```

```
fatherof(sheva,machbenah).
fatherof(ephrathah,hur).
fatherof(hur,shobal).
fatherof(hur,salma).
fatherof(hur,hareph).
fatherof(shobal,kiriath-jearim).
fatherof(salma,bethlehem).
fatherof(salma,atroth-beth-joab).
fatherof(hareph,beth-gader).
fatherof(shobal,haroeh).
fatherof(david,amnon).
fatherof(david,daniel).
fatherof(david,absalom).
fatherof(david,adonijah).
fatherof(david,shephatiah).
fatherof(david,ithream).
fatherof(david,shimea).
fatherof(david,shobab).
fatherof(david,nathan).
fatherof(david,solomon).
fatherof(david,ibhar).
fatherof(jeshaiah,rephaiah).
fatherof(rephaiah,arnan).
fatherof(arnan,obadiah).
fatherof(obadiah,shecaniah).
fatherof(shecaniah,shemaiah).
fatherof(shemaiah,hattush).
fatherof(shemaiah,igal).
fatherof(shemaiah,bariah).
fatherof(shemaiah,neariah).
fatherof(shemaiah,shaphat).
fatherof(neariah,elioenal).
fatherof(neariah,hizkiah).
fatherof(neariah,azriham).
fatherof(elioenai,hodaviah).
fatherof(elioenai,eliashib).
fatherof(elioenai,pelaiah).
fatherof(elioenai,akkub).
fatherof(elioenai,johahan).
fatherof(elioenai,delaiah).
fatherof(elioenai,anani).
fatherof(judah,perez).
fatherof(judah,hezron).
fatherof(judah,carmi).
fatherof(judah,hur).
fatherof(judah,shobal).
fatherof(shobal,reaiah).
fatherof(reaiah,jahath).
fatherof(jahath,ahumai).
fatherof(jahath,lahad).
fatherof(etam,jezreel).
fatherof(etam,ishma).
fatherof(etam,idbash).
fatherof(penuel,gedor).
fatherof(ezer,hushah).
fatherof(ashhur,tekoa).
fatherof(ashhur,ahuzam).
fatherof(josiah,shallum).
fatherof(jehoiakim,jeconiah).
fatherof(jeconiah,zedekiah).
fatherof(jeconiah,shealtiel).
fatherof(jeconiah,malchiram).
fatherof(jeconiah,pedaiah).
fatherof(jeconiah,shenazzar).
fatherof(jeconiah,jekamiah).
fatherof(jeconiah,hoshana).
fatherof(jeconiah,nedabiah).
fatherof(pedaiah,zerubbabel).
fatherof(pedaiah,shimei).
fatherof(zerubbabel,meshullam).
fatherof(zerubbabel,hananiah).
fatherof(zerubbabel,hashubab).
fatherof(zerubbabel,ohel).
fatherof(zerubbabel,berechiah).
fatherof(zerubbabel,hasadiah).
fatherof(zerubbabel,jushab-hesed).
fatherof(hananiah,pelatiah).
fatherof(hananiah,jeshaiah).
fatherof(eshton,paseah).
fatherof(eshton,tehinnah).
fatherof(tehinnah,ir-nahash).
fatherof(kenaz,othniel).
fatherof(kenaz,seraiah).
fatherof(othniel,hathath).
fatherof(othniel,meonothai).
fatherof(othniel,ophrah).
fatherof(seraiah,joab).
fatherof(joab,ge-harashim).
fatherof(jephunneh,caleb).
fatherof(caleb,iru).
fatherof(caleb,elah).
fatherof(caleb,naam).
fatherof(elah,kenaz).
fatherof(jehallelel,ziph).
fatherof(jehallelel,ziphah).
fatherof(jehallelel,tiria).
fatherof(jehallelel,asarel).
fatherof(ezrah,jether).
fatherof(ezrah,mered).
fatherof(ezrah,epher).
fatherof(ezrah,jalon).
fatherof(ezrah,ishbah).
fatherof(ishbah,eshtemoa).
fatherof(mered,shamai).
fatherof(mered,ishbah).
fatherof(mered,jered).
fatherof(mered,heber).
fatherof(mered,jekuthiel).
fatherof(jered,gedor).
fatherof(heber,soco).
fatherof(jekuthiel,zanoah).
fatherof(shimon,amnon).
fatherof(shimon,rinnah).
fatherof(shimon,ben-hanan).
```

```
fatherof(ashhur,hepher).
fatherof(ashhur,temeni).
fatherof(ashhur,ahashtari).
fatherof(ashhur,zereth).
fatherof(ashhur,zohar).
fatherof(ashhur,ethnan).
fatherof(koz,anub).
fatherof(koz,zobebah).
fatherof(chelub,mehir).
fatherof(mehir,eshton).
fatherof(eshton,bethrapha).
fatherof(laadah,mareshah).
fatherof(simeon,nemuel).
fatherof(simeon,jamin).
fatherof(simeon,jarib).
fatherof(simeon,zerah).
fatherof(simeon,shaul).
fatherof(simeon,shallum).
fatherof(simeon,mibsam).
fatherof(simeon,mishma).
fatherof(mishma,hammuel).
fatherof(mishma,shimei).
fatherof(mishma,zaccur).
fatherof(amaziah,joshash).
fatherof(amaziah,jamlech).
fatherof(amaziah,meshoabab).
fatherof(joshibiah,jehu).
fatherof(joshibiah,joel).
fatherof(seraiah,joshibiah).
fatherof(asiel,seraiah).
fatherof(shiphi,eiloenai).
fatherof(shiphi,jaakobah).
fatherof(shiphi,jeshohaiah).
fatherof(shiphi,asaiah).
fatherof(shiphi,adiel).
fatherof(shiphi,jesimiel).
fatherof(shiphi,benaiahi).
fatherof(shiphi,ziza).
fatherof(allon,shiphi).
fatherof(reuben,enoch).
fatherof(jedaiah,allon).
fatherof(shimri,jedaiah).
fatherof(shimri,shemaiah).
fatherof(reuben,pallu).
fatherof(reuben,hezron).
fatherof(reuben,carmi).
fatherof(joel,shemaiah).
fatherof(shemaiah,gog).
fatherof(gog,shimei).
fatherof(shimei,micah).
fatherof(micah,reaiah).
fatherof(reaiah,baal).
fatherof(baal,beerah).
fatherof(azaz,bela).
fatherof(shimon,tilon).
fatherof(ishi,zoheth).
fatherof(ishi,ben-zoheth).
fatherof(judah,shelah).
fatherof(shelah,er).
fatherof(shelah,laadah).
fatherof(shelah,jokim).
fatherof(shelah,joash).
fatherof(shelah,saraph).
fatherof(shelah,jahubilehem).
fatherof(er,lecah).
fatherof(jeshishai,michael).
fatherof(jahdo,jeshishai).
fatherof(buz,jahdo).
fatherof(abihail,michael).
fatherof(abihail,meshullam).
fatherof(abihail,sheba).
fatherof(abihail,jorai).
fatherof(abihail,jacan).
fatherof(abihail,zia).
fatherof(abihail,eber).
fatherof(abdiel,ahi).
fatherof(guni,abdiel).
fatherof(levi,gershom).
fatherof(levi,kohath).
fatherof(levi,merari).
fatherof(gershom,libni).
fatherof(gershomshimei).
fatherof(kohath,amram).
fatherof(kohath,izhar).
fatherof(kohath,hebron).
fatherof(kohath,uzziel).
fatherof(amram,aaron).
fatherof(amram,moses).
fatherof(amram,miriam).
fatherof(merari,mahli).
fatherof(merari,mushi).
fatherof(aaron,nadab).
fatherof(aaron,abihu).
fatherof(aaron,eleazar).
fatherof(aaron,ithamar).
fatherof(eleazar,phinehas).
fatherof(phinehas,abishua).
fatherof(abishua,bukki).
fatherof(bukki,uzzi).
fatherof(uzzi,zerahiah).
fatherof(zerahiah,meraioth).
fatherof(meraioth,amariah).
fatherof(amariah,ahitub).
fatherof(ahitub,zadok).
fatherof(zadok,ahimaaz).
fatherof(ahimaaz,azariah).
fatherof(azariah,johanan).
fatherof(johanan,azariah).
```

```
fatherof(huri,abihail).
fatherof(jaroah,huri).
fatherof(gilead,jaroah).
fatherof(michael,gilead).
fatherof(shallum,hilkiah).
fatherof(hilkiah,azariah).
fatherof(azariah,seraiah).
fatherof(seraiah,jehozadak).
fatherof(levi,gershom).
fatherof(levi,kohath).
fatherof(levi,merari).
fatherof(gershom,libni).
fatherof(gershom,shimei).
fatherof(kohath,amram).
fatherof(kohath,izhar).
fatherof(kohath,hebron).
fatherof(kohath,uzziel).
fatherof(merari,mahli).
fatherof(merari,mushi).
fatherof(libni,jahath).
fatherof(jahath,zimmah).
fatherof(zimmah,joah).
fatherof(joah,iddo).
fatherof(iddo,zerah).
fatherof(zerah,jeatherai).
fatherof(kohath,amminadab).
fatherof(kohath,korah).
fatherof(kohath,korah).
fatherof(kohath,assir).
fatherof(kohath,elkanah).
fatherof(kohath,ebiasaph).
fatherof(kohath,assir).
fatherof(kohath,tahath).
fatherof(kohath,uriel).
fatherof(kohath,uzziah).
fatherof(kohath,shaul).
fatherof(elkanah,amasai).
fatherof(elkanah,ahimoth).
fatherof(samuel,vashni).
fatherof(samuel,abijah).
fatherof(merari,mahli).
fatherof(mahli,libni).
fatherof(libni,shimei).
fatherof(shimei,uzzah).
fatherof(uzzah,shimea).
fatherof(shimea,haggiah).
fatherof(haggiah,asaiah).
fatherof(samuel,joel).
fatherof(elkanah,samuel).
fatherof(jeroham,elkanah).
fatherof(eliel,jeroham).
fatherof(azariah,amariah).
fatherof(amariah,ahitub).
fatherof(ahitub,zadok).
fatherof(zadok,shallum).
fatherof(toah,eliel).
fatherof(aaron,eleazar).
fatherof(eleazar,phinehas).
fatherof(phinehas,abishua).
fatherof(abishua,bukki).
fatherof(bukki,uzzi).
fatherof(uzzi,zerahiah).
fatherof(zerahiah,meraioth).
fatherof(meraioth,amariah).
fatherof(amariah,ahitub).
fatherof(ahitub,zadok).
fatherof(zadok,ahimaaz).
fatherof(issachar,tola).
fatherof(issachar,puah).
fatherof(issachar,jashub).
fatherof(issachar,shimron).
fatherof(tola,uzzi).
fatherof(tola,rephaiah).
fatherof(tola,jeriel).
fatherof(tola,jahmai).
fatherof(tola,ibsam).
fatherof(tola,shemuel).
fatherof(uzzi,izrahiah).
fatherof(izrahiah,michael).
fatherof(izrahiah,obadiah).
fatherof(izrahiah,joel).
fatherof(izrahiah,isshiah).
fatherof(benjamin,bela).
fatherof(benjamin,becher).
fatherof(benjamin,jediael).
fatherof(bela,ezbon).
fatherof(bela,uzzi).
fatherof(bela,uzziel).
fatherof(bela,jerimoth).
fatherof(bela,iri).
fatherof(becher,zemirah).
fatherof(becher,joash).
fatherof(becher,eliezer).
fatherof(becher,elioenai).
fatherof(becher,omri).
fatherof(becher,jeremoth).
fatherof(becher,abijah).
fatherof(becher,anathoth).
fatherof(becher,alemeth).
fatherof(jediael,bilhan).
fatherof(bilhan,jeush).
fatherof(bilhan,benjamin).
```

```
fatherof(bilhan,ehud).
fatherof(bilhan,chenaanah).
fatherof(bilhan,zethan).
fatherof(bilhan,tarshish).
fatherof(bilhan,ahishahar).
fatherof(ir,shuppim).
fatherof(ir,huppim).
fatherof(aher,hushim).
fatherof(naphtali,jahziel).
fatherof(naphtali,guni).
fatherof(naphtali,jezer).
fatherof(naphtali,shallum).
fatherof(manasseh,asriel).
fatherof(machir,gilead).
fatherof(machir,peresh).
fatherof(machir,sheresh).
fatherof(sheresh,ulam).
fatherof(sheresh,rekem).
fatherof(ulam,bedan).
fatherof(shemida,ahian).
fatherof(shemida,shechem).
fatherof(shemida,likhi).
fatherof(shemida,aniam).
fatherof(ephraim,shuthelah).
fatherof(shuthelah,bered).
fatherof(bered,tahath).
fatherof(tahath,eleadah).
fatherof(eleadah,tahath).
fatherof(tahath,zabad).
fatherof(zabad,shuthelah).
fatherof(zabad,ezer).
fatherof(zabad,elead).
fatherof(ephraim,beriah).
fatherof(rephah,resheph).
fatherof(resheph,telah).
fatherof(telah,tahan).
fatherof(tahan,ladan).
fatherof(ladan,ammihud).
fatherof(ammihud,elishama).
fatherof(elishama,non).
fatherof(non,joshua).
fatherof(asher,imnah).
fatherof(asher,ishvah).
fatherof(asher,ishvi).
fatherof(asher,beriah).
fatherof(asher,serah).
fatherof(beriah,heber).
fatherof(bela,huram).
fatherof(shaharaim,jobab).
fatherof(shaharaim,zibia).
fatherof(shaharaim,mesha).
fatherof(shaharaim,malcam).
fatherof(beriah,malchiel).
fatherof(malchiel,birzaith).
fatherof(heber,japhlet).
fatherof(heber,shomer).
fatherof(heber,hotham).
fatherof(heber,shua).
fatherof(japhlet,pasach).
fatherof(japhlet,bimhal).
fatherof(japhlet,ashvath).
fatherof(shemer,ahi).
fatherof(shemer,rohgah).
fatherof(shemer,hubbah).
fatherof(shemer,aram).
fatherof(helem,zophah).
fatherof(helem,imna).
fatherof(helem,shelesh).
fatherof(helem,amal).
fatherof(zophah,suah).
fatherof(zophah,harnepher).
fatherof(zophah,beri).
fatherof(zophah,shual).
fatherof(zophah,imrah).
fatherof(zophah,bezer).
fatherof(zophah,hod).
fatherof(zophah,shamma).
fatherof(zophah,shilshah).
fatherof(zophah,ithran).
fatherof(zophah,beera).
fatherof(jether,jephunneh).
fatherof(jether,pispa).
fatherof(jether,ara).
fatherof(ulla,arah).
fatherof(ulla,hanniel).
fatherof(ulla,rizia).
fatherof(benjamin,bela).
fatherof(benjamin,ashbel).
fatherof(benjamin,aharah).
fatherof(benjamin,nohah).
fatherof(benjamin,rapha).
fatherof(bela,addar).
fatherof(bela,gera).
fatherof(bela,abihud).
fatherof(bela,naaman).
fatherof(bela,abishua).
fatherof(bela,ahoah).
fatherof(bela,gera).
fatherof(bela,shephuphan).
fatherof(shashak,penuel).
fatherof(jeroham,shamsherai).
fatherof(jeroham,shehariah).
fatherof(jeroham,athaliah).
fatherof(jeroham,jaareshiah).
```

fatherof(shaharaim,jeuz).
fatherof(shaharaim,sachiah).
fatherof(shaharaim,mirmah).
fatherof(shaharaim,abitub).
fatherof(shaharaim,elpaal).
fatherof(elpaal,eber).
fatherof(elpaal,misham).
fatherof(elpaal,shemed).
fatherof(elpaal,beria).
fatherof(elpaal,shema).
fatherof(beriah,zebadiah).
fatherof(beriah,arad).
fatherof(beriah,eder).
fatherof(beriah,michael).
fatherof(beriah,ishpah).
fatherof(beriah,joha).
fatherof(elpaal,zebadiah).
fatherof(elpaal,meshullam).
fatherof(elpaal,hizki).
fatherof(elpaal,heber).
fatherof(elpaal,ishmeraj).
fatherof(elpaal,izliah).
fatherof(elpaal,jobab).
fatherof(shimei,jakim).
fatherof(shimei,zichri).
fatherof(shimei,zabdi).
fatherof(shimei,elienai).
fatherof(shimei,zillethai).
fatherof(shimei,eliel).
fatherof(shimei,adaiah).
fatherof(shimei,beraiah).
fatherof(shimei,shimrath).
fatherof(shashak,ishpan).
fatherof(shashak,eber).
fatherof(shashak,eliel).
fatherof(shashak,abdon).
fatherof(shashak,zichri).
fatherof(shashak,hanan).
fatherof(shashak,hananiah).
fatherof(shashak,elam).
fatherof(shashak,anthothiah).
fatherof(shashak,iphdeiah).
fatherof(reuel,shephatiah).
fatherof(ibneiah,reuel)
fatherof(hilkiah,azariah).
fatherof(meshullam,hilkiah).
fatherof(zadok,meshullam).
fatherof(meraioth,zadok).
fatherof(ahitub,meraioth).
fatherof(jeroham,elijah).
fatherof(jeroham,zichri).
fatherof(mikloth,shimeah).
fatherof(ner,kish).
fatherof(kish,saul).
fatherof(saul,jonathan).
fatherof(saul,malchi-shua).
fatherof(saul,abinadab).
fatherof(saul,eshbaal).
fatherof(jonathan,merib-baal).
fatherof(merib-baal,micah).
fatherof(micah,pithoh).
fatherof(micah,melech).
fatherof(micah,taarea).
fatherof(micah,ahaz).
fatherof(ahaz,jehoaddah).
fatherof(jehoaddah,alemeth).
fatherof(jehoaddah,azmaveth).
fatherof(jehoaddah,zimri).
fatherof(zimri,moza).
fatherof(moza,binea).
fatherof(binea,eleasah).
fatherof(eleasah,azel).
fatherof(azel,azrikam).
fatherof(azel,bocheru).
fatherof(azel,ishmael).
fatherof(azel,sheariah).
fatherof(azel,obadiah).
fatherof(azel,hanan).
fatherof(eshek,ulam).
fatherof(eshek,jeush).
fatherof(eshek,eliphelet).
fatherof(ammihud,uthai).
fatherof(omri,ammihud).
fatherof(zerah,jeuel).
fatherof(meshullam,sallu).
fatherof(hodaviah,meshullam).
fatherof(hassenuah,hodaviah).
fatherof(jeroham,ibneiah).
fatherof(uzzi,elah).
fatherof(michri,uzzi).
fatherof(shephatiah,meshullam).
fatherof(jeroham,adaiah).
fatherof(pashhur,jeroham).
fatherof(malchijah,pashhur).
fatherof(adiel,maasai).
fatherof(jahzerah,adiel).
fatherof(meshullam,jahzerah).
fatherof(meshillemith,meshullam).

参 考 文 献

第一节 数 学 史

一、关于数学史的原始资料

Birkhoff, G. , ed. , *A Source Book in Classical Analysis*, Harvard University Press, Cambridge, Mass. , 1973.

本书描绘了分析学变得精确的历史。

Heinzmann, G. , ed. , *Poincaré, Russell, Zermelo et Peano: textes de la discussion (1906-1912) sur les fondements des mathématiques: des antinomies a la predicativite*, Albert Blanchard, Paris, 1986.

Midonick, D. , *The Treasury of Mathematics*, Philosophical Library, New York, 1965.

Smith, D. E. , *A Source Book in Mathematics*, 2 vols. , Dover, New York, 1959.

Struik, D. , *A Source Book in Mathematics 1200-1800*, Harvard University Press, Cambridge, Mass. , 1969.

这个时期很多数学家的代表性论文。

Thomas, I. , *Selections Illustrating the History of Greek Mathematics with an English Translation*, 2 vols. , Loeb Classical Library, Harvard University Press, Cambridge, Mass. , 1939.

一本希腊语–英语对照版本的选集，Heath 在他的《History of Greek Mathematics》中引用了它，并加以丰富的评注。

二、数学史

Asprey, W. and Kitcher, P. , *History and Philosophy of Modern Mathematics*, Minnesota Studies in the Philosophy of Science, vol. XI, University of Minnesota Press, Minneapolis, 1988.

一本中立的文选，致力于展现当时各学科之间的活动情况。

Boyer, C. , *A History of Mathematics*, 2nd ed. , Revised by U. C. Merzbach, Wiley, New York, 1989.

Browder, F. E. , ed. , *Mathematical developments arising from Hilbert's problems*, *Proc. Symp. Pure Math.* , **27**, American Mathematical Society, Providence, 1976.

Cantor, G. , "Über die Ausdennung eines Satzes aus der Theorie der trigonomerischen Reihen", *Mathematische Annalen*, **5**, 123-132, 1872.

Dieudonné, J. , *Abrégé d'Histoire des Mathématiques*, 2 vols. , Hermann, Paris, 1978.

DuGac, P. , "Éléments d'analyse de Karl Weierstrass", *Archive for the History of the Exact Sciences*, **10**, 42-176, 1973.

DuGac, P. , *Richard Dedekind et les fondements de l'analyse*, Paris, 1976.

Edwards, C. H. , *The Historical Development of the Calculus*, Springer-Verlag, Berlin, 1979.

Eves, H. , *An Introduction to the History of Mathematics*, 4th ed. , Holt Rinehart Winston, New York, 1975; 6th ed. , Saunders College Pub. , Philadelphia, 1990.

Grattan-Guinness, I. , *The Development of the Foundations of Mathematical Analysis from Euler to Riemann*, MIT Press, Cambridge, Mass. , 1970.

说明了欧拉、高斯和黎曼在当时坚固的数学基础上建立自己的数学时所遇到的困难。

Grattan-Guinness, I. , ed. , *From the Calculus to Set Theory 1630-1910: An Introductory History*, Duckworth, London, 1980.

Heath, T . L. , *Mathematics in Aristotle*, Clarendon Press, Oxford, 1949.

Heath, T, L. , *A History of Greek Mathematics I, II*, Dover, New York, 1981.

上面的这两本书是这个领域的标准教材。

Heine, E. , "Die Elemente der Functionenlehre", *J. fur die reine und angewandte Mathematik*, **74**, 172-188,

1872.

Kitcher, P. , *The Nature of Mathematical Knowledge*, Oxford University Press, Oxford, 1983.

除了大量对哲学问题的处理之外，这本书还以一段现代分析学的历史为案例，来说明作者的观点。

Klein, F. , *Development of Mathematics in the* 19^{th} *Century* (M. Ackermann, tr.), Math. Sci. Press, Brookline, Mass, 1979.

迄今为止关于 19 世纪核心数学演变的最好阐述，清晰地表明了数学的及其基础的困难所在。

Kolmogorov, A. N. and Yushkevich, A. P. , eds. , *Mathematics of the 19th Century: Mathematical Logic, Algebra, Number Theory, Probability Theory* Birkhäuser, Boston, 1992.

Philips, E. , ed. , *Studies in the History of Mathematics*, MAA Studies in Mathematics, **26**, Mathematical Association of America, 1987.

Smith, D . E. , *A History of Mathematics*, 2 vols. , Dover, New York, 1958.

Stillwell, J. , *Mathematics and its History*, Springer-Verlag, 1989.

Struik, D. J. , *A Concise History of Mathematics*, Dover, New York, 1987.

第二节 逻辑史

逻辑学的历史同数学史一样是和哲学史紧密联系在一起的。下面的参考书反映了这一点。

一、逻辑学的原始资料

这里有五本关于现代数理逻辑及其应用的奠基性的论文集，可以通过查阅这些论文集澄清一些我们未解释的内容。学完本书前两章的学生有足够的能力阅读大多数论文。而这些论文也可以作为教师教学的阅读材料和论文作业。

Benacerraf, P. and Putnam, H. , eds. , *Philosophy of Mathematics: Selected Readings*, 2^{nd} ed. , Cambridge University Press, Cambridge, 1983.

为那些希望了解哲学问题的人而写。

Davis, M. , ed. , *The Undecidable. Basic papers on undecidable propositions, unsolvable problems, and computable functions*, Raven Press, Hewlett, N. Y. , 1965.

这一卷包括了 1930 年至 1940 年间，由哥德尔、丘奇、克林、波斯特和图灵撰写的关于递归函数论、不完全性和不可判定性的论文。

van Heijenoort, J. , ed. , *From Frege to Gödel*, Harvard University Press, Cambridge, Mass. , 1967.

在这本书中可以看到 1879 年到 1931 年间数理逻辑方面最重要的论文(英语版本)，包括大多数在历史附录中所提到的。顶尖逻辑学家对这些论文所做的介绍对于理解论文，尤其是晦涩难懂的论文，是非常有益的。

Hintikka. , J. , ed. , *The Philosophy of Mathematics* (Oxford Readings in Philosophy), Oxford University Press, London, 1969.

Siekmann, J. and Wrightson, G. , eds. , *Automation of Reasoning*, 1957-70, 2 vols. , Springer-Verlag, Berlin, 1983.

这一卷包括大部分关于自动推理的早期论文，其中许多论文都可以作为很好的课后阅读材料。

二、逻辑史

Bochenski, J. , *Ancient Formal Logic*, North-Holland, Amsterdam, 1951.

Bochenski, J. , *A History of Formal Logic*, University of Notre Dame Press, Notre Dame, Ind. , 1961.

Boehner, P. , *Medieval Logic*, Manchester University Press, Manchester, England, 1952.

Drucker, T. , ed. , *Perspectives on the History of Mathematical Logic*, Birkhäuser, Boston, 1991.

Hailperin, T. , *Boole's logic and Probability*, North-Holland, Amsterdam, 1976.

Hallet, Michael, *Cantorian Set Theory and Limitation of Size* (Oxford Logic Guides, **10**), Clarendon Press, Oxford, 1984.

Kanamori, A., "The mathematical development of set theory from Cantor to Cohen", *B. Symbolic Logic*, 2, 1-71, 1996.

一篇精彩的短文，广泛地叙述了科恩的工作之前的集合论历史。

Kneale, W. and Kneale, M., *The Development of Logic*, Clarendon Press, Oxford, 1975.

标准的阐述。

Lukasiewicz, J., *Aristotle's Syllogistic from the Standpoint of Modern Formal Logic*, 2nd ed., Clarendon Press, Oxford, 1957.

从另一个角度看亚里士多德的工作。

Mates, B., *Stoic Logic*, 2nd ed., University of California Press, Berkeley, 1961.

Mueller, I., *Philosophy of Mathematics and Deductive Structure in Euclid's Elements*, MIT Press, Cambridge, Mass., 1981.

Nidditch, P,, *The Development of Mathematical Logic*, Routledge and Kegan Paul, London, 1962.

Parkinson, G., *Logic and Reality in Leibniz's Metaphysics*, Clarendon Press, Oxford, 1965.

可能是学习莱布尼茨逻辑的最好的材料。

R escher, N., *The Development of Arabic Logic*, 1964.

Schilpp, P., *The Philosophy of Bertrand Russell*, Northwestern University, Evanston, Ill., 1944 (3rd ed., Tudor Publishers, New York, 1951).

Scholz, H., *Concise History of Logic* (K. Leidecker, tr.), Philosophical Library, New York, 1961.

Styazhkin, N. I., *History of Mathematical Logic, from Leibniz to Peano*, MIT Press, Cambridge, Mass., 1969.

三、原始的逻辑历史资料

Aristotle, *Selections*(W. D. Ross, ed.), Oxford University Press, Oxford, 1942.

Aristotle, *Categories and De Interpretatione*(J. Ackrill, tr.), Clarendon Press, Oxford, 1966.

Aristotle, *Aristotle's Posterior Analytics*(J. Barnes, tr.), Clarendon Press, Oxford, 1975.

Aristotle, *Aristotle's Categories and Propositions*(H. Apostle, tr.), Peripatetic Press, Grinnell, Iowa, 1980.

Bernays, P., *A system of Axiomatic Set Theory, Sets and Classes: On the Work by Paul Bernays*, (G. Müller, ed.), North-Holland, Amsterdam, 1976.

包括了贝尔奈斯(Bernays)1941 年至 1954 年间在《Journal of Symbolic Logic》上发表的系列论文。

Boole, G., *The Mathematical Analysis of Logic*, Macmillan, Barclay and Macmillan, Cambridge, England, 1847(reprinted B. Blackwell, Oxford, 1948).

十分有趣的读物。

Boole, G., *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probability*, Macmillan, London, 1854.

Boole, G., *Collected Logical Works*, Open Court, La Salle, Ill., 1952.

Boole, G., *Studies in Logic and Probability*, Watts, London, 1952.

Boole, G., *A Treatise on Differential Equations*, 5th ed. Chelsea, New York, 1959.

Boole, G., *A Treatise on the Calculus of Finite Differences*, J. F. Moulton, ed., 5th ed., Chelsea, New York, 1970.

Boole, G., *The Boole-De Morgan Correspondence 1842-1864* (G. C. Smith, ed.), Clarendon Press, Oxford, 1982.

Brouwer, L. E. J., *Collected Works*, 2 vols. (A. Heyting, ed.), American Elsevier, New York, 1975.

Cantor, G., "Rezension der Schrift von G. Frege, *Die Grundlagen der Arithmetik*", *Deutsche Littertur Zeitung*, 728-729, 1885 (reprinted in Cantor[1932, 2.3], 440-441).

Cantor, G., *Gesammelte Abhandlungen mathematischen und philosophischen Inhalts* (E. Zermelo, ed.), Springer, Berlin, 1932 (reprinted 1962, Olms, Hildesheim).

Cantor, G., *Briefwechsel Cantor-Dedekind* (E. Noether and J. Cavaillès, eds.), Actualitates scientifiques et industrielles, **518**, Hermann, Paris, 1937.

Cantor, *Contributions to the Founding of the Theory of Transfinite Numbers* (P. E. B. Jourdain, tr.), Dover, New York, 1952.

可以看到康托的工作基础是如何地非形式和直觉的。

Cavaillès, J., *Philosophie mathématique*, includes "Correspondence Cantor-Dedekind", Hermann, Paris, 1962.

Couturat, L., *Opuscules et Fragments inédits de Leibniz, extracts des manuscripts de la Bibliothéque royale de Hanovre*, Paris, 1903.

Couturat, L., *L'Algebre de la Logique*, 2nd ed., A. Blanchard, Paris, 1980 (translation of original edition by L. G. Robinson as *The Algebra of Logic*, Open Court, London, 1914).

De Morgan, A., *Formal Logic, or, the calculus of inference, necessary and probable*, Taylor and Walton, London, 1847.

Dedekind, R., *Gesammelte mathematische Werke*, 3 vols. (R. Fricke, E. Noether and O. Ore, eds.), F. Vieweg & Sohn, Brunswick, 1932 (reprinted Chelsea, New York, 1969).

Dedekind, R., *Essays on the Theory of Numbers* (W. Beman, tr.), Dover, New York, 1963.

戴德金是具有独特风格的大师。这些通过戴德金分割构造实数和通过归纳定义函数的文章是明晰易懂的杰作。

Descartes, R., *Discours de la Methode*, Ian Marie, Leyde, 1637 (reprinted Union generale d'editions, Paris, 1963; translated as *Discourse on Method* by J. Veitch, Open Court, La Salle, Ind. 1945).

Euclid, *The thirteen books of Euclid's Elements* (Sir Thomas Heath, tr. and ed.), Dover, New York, 1956.

Fraenkel, A., "Der Begriff 'definit' und die Unabhängigkeit des Auswahls-axioms", *Sitzungsberichte der Preussischen Akademie der Wissenchaften, Physikalisch-mathematische Klasse*, 253-257, 1922 (translated in van Heijenoort [1967, 2.1] as "The notion of 'definite' and the independence of the axiom of choice").

Fraenkel, A., "Zu den Grundlagen der Cantor-Zermeloschen Mengenlerhe", *Math. Annalen*, **86**, 230-237, 1922a.

Frege, G., *Begriffsschrift*, Halle, 1879 (translated in van Heijenoort [1967, 2.1] as "*Begriffsschrift*, a formula language, modeled upon that of arithmetic, for pure thought").

Frege, G., *Die Grundlagen der Arithmetik*, Breslau, 1884 (reprinted and translated as *The Foundations of Arithmetic* by J. L. Austin, Philosophical Library, New York, 1953).

Frege, G., *Grundgesetze der Arithmetik, begriffsschriftlich abgeleitet*, 2 vols., Jena, Pohl, 1903 (partially translated by M. Furth as *The basic laws of arithmetic. Exposition of the System*, University of California Press, Berkeley, 1964).

Frege, G., *Conceptual notation and Related articles* (T. W. Bynum, tr. and ed.), Clarendon, Oxford, 1972.

Frege, G., *Logical Investigations* (P. T. Geach, ed., P. T. Geach and R. H. Stoothoff, tr.), R. Blackwell, Oxford, 1977.

这些名著至今还有许多值得学习的地方。

Gentzen, G. (M. E. Szabo, ed.), *The Collected Papers of Gerhard Gentzen*, Studies in Logic, North-Holland, Amsterdam, 1969.

作者比他的大多数继承者更好地解释了自己的工作。

Gödel, K., "Die Vollständigkeit der Axiome des logischen Funktionenkalküls", *Mon. für Math. und Physik*, **37**, 349-360, 1930 (translated as "The completeness of the axioms of the functional calculus of logic", in van Heijenoort [1967, 2.1] and Gödel [1986, 2.3]).

Gödel, K., "Über formal unentscheidbare…", *Mon. für Math. und Physik*, **37**, 349-360, 1931 (translated as "On formally undecidable propositions of Principia Mathematica and related systems I", in van Heijenoort [1967, 2.1], Davis ([1965, 2.1] and Gödel ([1986, 2.3]).

Gödel, K., "The consistency of the Axiom of Choice and of the Genralized Continuum Hypothesis", *Proc. Nat. Ac. Sci.*, **24**, 556-557, 1938 (reprinted in Gödel [1986, 2.3] vol. 2, 26-27).

Gödel, K., "Consistency-proof for the Genralized Continuum Hypothesis", *Proc. Nat. Ac. Sci.*, **25**, 220-224, 1939 (reprinted in Gödel [1986, 2.3] vol. 2, 28-32).

Gödel, K., "Eine Interpretation des intuitionistischen Aussagenkalküls", Ergebnisse eines mathematischen Kolloquiums, 4, 1933 (reprinted and translated as "An interpretation of the intuitionistic propositional calculus", in Gödel [1986, 2.3] vol. 1, 300-302.

Gödel, K., "Russell's mathematical logic", in Schilpp [1944, 2.2], 123-153 (reprinted in Gödel [1986, 2.3] vol. 2, 119-141).

Gödel, K., *Collected Works*, vol. 1 - (S. Feferman et al., eds.), Oxford University Press, Oxford, 1986 - .
专家给这些文章撰写的介绍和解释，在其他地方是找不到的。

Herbrand, J., *Recherches sur la théorie de la démostration*, Thesis, University of Paris, 1930 (excerpts translated as "Investigations in proof theory: The properties of true propositions" in van Heijenoort [1967, 2.1]).

Herbrand, J., *Logical Writings*, Reidel, Hingham, Mass., 1971.

Heyting, A., *Intuitionism: An Introduction*, North Holland, Amsterdam, 1956.

Hilbert, D., *Grundlagen der Geometrie*, Teubner, Leipzig, 1899 (translated by E. J. Townsend as *Foundations of Geometry*, Open Court, La Salle Ill., 1950).
第一个对欧氏几何进行了完整阐述，本书的英文版补充了一些重要的内容。

Hilbert, D. and Ackermann, W., *Grundzüge der theoretischen Logik*, Springer, Berlin, 1928 (English translation of 1938 edition, Chelsea, New York, 1950).

Hilbert, D. and Bernays, P., *Grundlagen der Mathematik I(1934) II (1939)*, 2nd ed., I (1968) II (1970), Springer, Berlin.
未被翻译，却是数学基础方面影响力最大的书。

Löwenheim, L., "Über Möglichkeiten im Relativkalkül", *Math. Annalen*, **76**, 447-470, 1915 (translated as "On possibilities in the calculus of relatives" in van Heijenoort [1967, 2.1].

Peano, G., *Formulario mathematico*, Bocca Freres, Turin, 1894-1908 (reprinted Edizioni Cremonese, Rome, 1960).

Peano, G., *Opere Scelte*, 3 vols., Edizioni Cremonese, Rome, 1957-9.

Peano, G., *Selected Works of Giuseppe Peano* (H. C. Kennedy, tr.), University of Toronto Press, Toronto, 1973.

Peano, G., *Arbeiten zur analysis und zur mathematischen Logik* (edited by G. Asser), B. G. Teubner, Leipzig, 1990.
皮亚诺的文章具有很强的可读性，因为我们今天大量采用了他提出的符号（除了把他的点表示法改为括号）。

Peirce, C. S., "Description of a notation for the logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic", *Memoirs of the American Academy of Arts and Sciences*, **9**, 317-368, 1870 (reprinted in Peirce [1933, 2.3], 27-98).

Peirce, C. S., *Collected Papers of Charles Sanders Peirce*, C. Hartshorne and P. Weiss, eds., vol. 3, *Exact Logic*, The Belknap Press of Harvard University Press, Cambridge, Mass., 1933 (reprinted in 1960).
皮尔斯独立发明了谓词逻辑和关系演算。

Poincaré, H., *Science and Hypothesis* (W. J. Greenstreet, tr.), Dover, New York, 1952.

Poincaré, H., *Science and Method* (F. Maitland, tr.), Dover, New Fork, 1952.

Post, E., "Introduction to a general theory of propositions", *American Journal of Mathematics*, **43**, 163-185, 1921 (reprinted in van Heijenoort [1967, 2.1]).

Russell, B., *The Principles of Mathematics*, Cambridge University Press, Cambridge, England, 1903.
一本非形式化的并且写得极好的书，带有那个时代的风格。

Russell, B. , On some difficulties in the theory of transfinite numbers and order types, *Proc. London Math. Soc.* (2), **4**, 29-53, 1907.

Russell, B. , *Introduction to Mathematical Philosophy*, Allen and Unwin, London, 1917 (reprinted 1960).

逻辑观点的极好论述。

Russell, B. , *Logic and Knowledge: Essays 1901-1950* (R. C. Marsh, ed.), Allen and Unwin, London, 1956.

Russell, B. , *The Autobiography of Bertrand Russell*, 3 vols. , Allen and Unwin, London, 1967.

并未涉及太多的逻辑，但是揭示了一些社会背景。

Schröder, E. , *Der Operationskreis des Logikkalkulus*, Leipzig, 1877 (reprinted Darmsdat, 1966).

Schröder, E. , *Vorlesungen über die Algebra der Logik*, 3 vols. , B. G. Teubner, Leipzig, 1890-1905.

Skolem, T. , "Einge Bemerkungen zur axiomatischen Begründung der Mengenlehre" in *Matematikerkongressen i Helsingfors*, Akademiska Bokhandeln, Helsinki, 1922 (translated in van Heijenoort [1967, 2. 1] as "Some remarks on axiomatized set theory").

Whitehead, A. N. and Russell, B. , *Principia Mathematica*, 3 vols. , Cambridge University Press, Cambridge, England, 1910-13; 2nd ed. , 1925-1927.

引用数比阅读数还大的书。

Wittgenstein, L. , *Tractatus Logico-Philosophicus* (D. Pears and B. McGuiness, tr.), Routledge and Kegan Paul, London, 1974.

年轻的维特根斯坦比我们更严谨地对待真值表。

von Neumann, J. , "Zur Einführung der transfiniten Zahlen", *Acta literarum ac scientiarum Regiae Universitatis Hungaricae Francisco-Josephinae, Sectio scientarum mathematicarum*, **1**, 199-208, 1923 (translated as "On the introduction of transfinite numbers" in van Heijenoort [1967, 2. 1].

von Neumann, J. , Eine Axiomatisierung der Mengenlehre, *J. für die reine und angewandte Mathematik*, **154**, 219-240, 1925 (translated as "An axiomatization of set theory" in van Heijenoort [1967, 2. 1]).

von Neumann, J. , *Collected Works*, vol. 1 (A. H. Taub, ed.), Pergamon Press, Oxford, 1961.

Zermelo, E. , "Beweis, dass jede Menge wohlgeordnet werden kann", *Math. Annalen*, **59**, 514-516, 1904 (translated as "Proof that every set can be well-ordered" in van Heijenoort [1967, 2. 1]).

Zermelo, E. , "Untersuchungen über die Grundlagen der Mengenlehre I, *Math. Annalen*, **65**, 261-281, 1908 (translated as "Investigations into the foundations of set theory I" in van Heijenoort [1967, 2. 1]).

第三节 数理逻辑

一、手册

(另见列表第五节一)

Barwise, J. ed. , *Handbook of Mathematical Logic*, North Holland, Amsterdam 1977.

集体合作完成，用一种相当独立的方式总结了现代数理逻辑的各个方向，是了解不熟悉的逻辑分支的首选。

Gabbay, D. and Guenthner, F. , *Handbook of Philosophical Logic*, 4 vols. , D. Reidel, Dordrecht, 1983-85.

这本书对非经典逻辑许多分支做了非常有用的总结。

二、逻辑学教材

(另见列表第五节二)

Beth, E. , *Formal Methods; An Introduction to Symbolic Logic and to the Study of Effective Operations in Arithmetic and Logic*, D. Reidel, Dordrecht, 1962.

Beth, E. , *The Foundations of Mathematics: A Study in the Philosophy of Science*, 2nd ed. , North-Holland, Amsterdam, 1965.

这本书广泛地探讨了数学基础及历史。

Boolos, G. and Jeffrey, R., *Computability and Logic*, 3rd ed., Cambridge University Press, Cambridge, England, 1989.

Church, A. *Introduction to Mathematical Logic*, rev. ed., Princeton University, Press, Princeton, 1956.

这本书广泛地探讨了经典命题与谓词逻辑的另一种公理系统。

Crossley, J. N. et al., *What is Mathematical Logic?*, Oxford University, Press, Oxford, 1972 (reprinted Dover, New York, 1990).

作为不完全性的简要介绍，这本书被翻译成了多种语言。

Curry, H. B., *Foundations of Mathematical Logic*, Dover, New York, 1977.

形式主义者感兴趣的读物。

Ebbinghaus, H., Flum, J. and Thomas, W., *Mathematical Logic*, Springer-Verlag, Berlin, 1984.

可读性强，而且是一本很好的基础教材。

Enderton, H., *A Mathematical Introduction to Logic*, Academic Press, New York, 1972.

Hamilton, A. G., *Logic for Mathematicians*, Cambridge University Press, Cambridge, England, rev. ed., 1988.

Hilbert, D. and Ackermann, W., *Principles of Mathematical Logic* (L. Hammond et al. tr. of [1928, 2.3].), Chelsea, New York, 1950.

现代第一本关于一阶逻辑的著作，因其表述简短清晰而著称于世。

Hodges, W., *Logic*, Penguin, Harmondsworth, England, 1977.

Kalish, D., Montague, R. and Mar, G., *Techniques of Formal Deduction*, Harcourt Brace Jovanovich, New York, 1980.

Keisler, H. J. and Robbin, J., *Mathematical Logic and Computability*, McGraw-Hill, New York, 1996.

威斯康星大学麦迪逊分校(University of Wisconsin at Madison)的逻辑小组新编写的本科生教材，包含了一个计算机程序包。

Kleene, S. C., *Introduction to Metamathematics*, D. Van Nostrand, New York, 1952 (reprinted North-Holland, Amsterdam, 1971).

这本书通过递归论的主要构成部分对逻辑进行了介绍，从而区分了直觉主义和经典的成分，并且对哥德尔定理和相关话题作了不可超越的解释。

Lukasiewicz, J., *Elements of Mathematical Logic* (O. Wojtasiewicz, tr.), MacMillan, New York, 1963.

Manaster, A., *Completeness, Compactness, Undecidability: An Introduction to Mathematical Logic*, Prentice-Hall, Englewood Cliffs, N. J., 1975.

这本教材对矢列式计算作了简要的说明。

Mendelson, E., *Introduction to Mathematical Logic*, 2nd ed., D. Van Nostrand, New York, 1979.

30 年来，专业逻辑学家使用最多的本科生教材。

Monk, J. D., *Mathematical Logic*, GTM **37**, Springer-Verlag, Berlin, 1976.

Ponasse, D., *Mathematical Logic*, Gordon Breach, New York, 1973.

Quine, W, V., *Mathematical Logic*, rev. ed., Harvard University Press, Cambridge, Mass., 1951.

奎因的哲学观启发出的高阶逻辑和集合论的概要。

Rosenbloom, P. C., *The Elements of Mathematical Logic*, Dover, New York, 1950.

这本教材很好地介绍了波斯特正则系统，并按照波斯特的做法把它用到不完全性的证明。

Rosser, J. B., *Logic for Mathematicians*, 2nd ed., Chelsea, New York, 1978.

这本教材以奎因的 NF 系统作为基础。

Shoenfield, J. R., *Mathematical Logic*, Addison-Wesley, Reading, Mass., 1967.

这本教材对 1967 年左右的数理逻辑作了均衡的概括，至今仍然是研究生逻辑课程的基本参考书。有很多极好的习题。不过这本书写得非常简要。

Smullyan, R., *First Order Logic*, Springer-Verlag, Berlin, 1968.

这本书给出了多种版本的表方法，并用它们与相容性推导出大量的结果，这些结论没有在本书中提到。

Smullyan, R., *What is the Name of this Book?*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

Smullyan, R., *Forever Undecided: a Puzzle Guide to Gödel*, A. Knopf, New York, 1987.

上面两本是 Smullyan 所著的流行的难题书，以很严密的方式和与众不同的术语处理哥德尔的定理。

van Dalen, D., *Logic and Structure*, 2nd ed., Springer-Verlag, Berlin, 1983.

这本教材以优美的文字介绍了直觉主义逻辑和经典逻辑。

三、集合论

Cohen, P., "The independence of the Continuum Hypothesis I", *Proc. Nat. Ac. Sci.*, **50**, 1143-1148, 1963.

Cohen, P., "The independence of the Continuum Hypothesis II", *Proc. Nat. Ac. Sci.*, **51**, 105-110, 1964.

Cohen, P., *Set Theory and the Continuum Hypothesis*, W. A. Benjamin, Inc., New York, 1966.

这本书给出了作者关于选择公理和连续统假设的著名的独立性证明，并对逻辑做了一个简要精彩的介绍。

Devlin, K. J., *The Joy of Sets: Fundamentals of Contemporary Set Theory*, 2nd ed., Springer-Verlag, Berlin, 1993.

Drake, F. R., *Set Theory: An Introduction to Large Cardinals*, North-Holland, Amsterdam, 1974.

关于大基数的早期教材。

Easton, W. *Powers of Regular Cardinals*, Ph. D. Thesis, Princeton University 1964. (An abridged version appears in *Ann. Math, Logic*, **1**, 139-178.)

Fraenkel, A. A. and Bar-Hillel, Y., *Foundations of Set Theory*, North-Holland, Amsterdam, 1958 (2nd ed., with D. van Dalen, 1973).

本书完整地描述了集合论各公理的本质和来源。

Fraenkel, A. A., *Abstract Set Theory*, 4th ed., revised by A. Levy, North-Holland, Amsterdam, 1976.

Gödel, K., *The Consistency of the Axiom of Choice and of the Generalized Continuum Hypothesis with the axioms of Set Theory*, *Ann. Math. Studies* **3**, Princeton University Press, Princeton, 1940 (reprinted in Gödel [1986, 2.3] vol. 2, 33-101).

Halmos, P., *Naive Set Theory*, D. Van Nostrand, Princeton, 1960.

Hausdorff, F., *Set Theory*(tr. from German 3rd edition, 1937), Chelsea, New York, 1962.

尽管早在 1914 年就首次出版，这本书仍然是一本十分重要的关于集合论方法在拓扑学上应用的参考书。

Hrbaček, K. and Jech, T., *Introduction to Set Theory*, 2nd rev. ed., M. Dekker, New York, 1984.

Jech, T., *The Axiom of Choice*, North Holland, Amsterdam, 1973.

收集了许多形式的选择公理的独立性证明。

Jech, T., *Set Theory*, Academic Press, New York, 1978.

研究生水平的标准参考书，覆盖了集合论的基本内容。

Jech, T., "Singular cardinals and the PCF theory", *B. Symbolic Logic*, **1**, 408-424, 1995.

一篇关于近期基数算术工作的很好的综述(强调了谢拉赫的工作)。

Kanamori, A. *The Higher Infinite*, Springer-Verlag, Berlin, 1994.

现行的高级教材，涵盖了大基数、描述集合论和确定性公理等诸多主题。另外还在参考书目中对这些领域近期的发展作了历史回顾。

Kechris, A. S., *Classical Descriptive Set Theory*, Springer-Verlag, Berlin, 1995.

描述了关于实数和相关拓扑空间的集合论，同时加入了该领域的各种方法。

Kunen, K., *Set Theory: An Introduction to Independence Proofs*, North-Holland, Amsterdam, 1980.

课堂内最受欢迎的研究生基础教材。

Kuratowski, K. and Mostowski, A., *Set Theory*, North-Holland, Amsterdam, 1976.

Levy, A., *Basic Set Theory*, Springer-Verlag, Berlin, 1979.

Moschovakis, Y., *Descriptive Set Theory*, North-Holland, Amsterdam, 1980.

多年来关于实数集合论及其逻辑观点的推广的标准教材。

Moschovakis, Y., *Notes on Set Theory*, Springer-Verlag, Berlin, 1994.

Quine, W. V., *Set Theory and Its Logic*, Belknap Press of Harvard University Press, Cambridge Mass., rev. ed. 1969.

介绍了奎因(Quine)的哲学观(主要是本体论)所推动的基础集合论。还对一些主要的集合论系统作了比较，例如罗素的类型论和奎因自己的新基础论。

Rubin, H. and Rubin, E., *Equivalents of the Axiom of Choice, I, II*, North-Holland, Amsterdam, 1963, 1985.

Rubin, J., *Set Theory for the Mathematician*, Holden-Day, San Francisco, 1967.

Shelah, H., *Cardinal Arithmetic*, Oxford Logic Guides, Oxford University Press, Oxford, 1994.

Sierpinski, W., *Cardinal and ordinal numbers*, PWN, Warsaw, 1965.

清晰的讲解，仍然值得一读。

Silver, J., "On the singular cardinals problem", *Proc. Intl. Cong. Math.*, R. D. James ed., Canadian Mathematical Congress, Montreal, 1974, 265-268.

Vaught. R., *Set Theory: An Introduction*, 2nd ed., Birkhauser, Boston, 1994.

四、模型论

Barwise, J. and Feferman, S., eds., *Model-Theoretic Logics*, Springer-Verlag, Berlin, 1985.

巨著，主要是对逻辑和模型论的概括，并不局限于一阶逻辑。

Bell, J. L. and Slomson, A. B., *Models and Ultraproducts*, North-Holland, Amsterdam, 1974.

被许多学生公认为是最好懂的入门读物。

Chang, C. C. and Keisler, H. J., *Model Theory*, 3rd ed., North-Holland, Amsterdam, 1990.

仍然是基础模型论最标准的论著，两位作者都是这个领域中主要的贡献人。

Henkin, L., "The completeness of the first order functional calculus", *J. Symbolic Logic*, **14**, 159-166, 1949.

Hodges, W., *Model Theory*, Cambridge University Press, Cambridge, 1993.

对当前模型论各方向的一个广泛的概览，应该会成为标准的教材。

Maltsev, A., *The Metamathematics of Algebraic Systems* (B. F. Wells III, tr., ed.) North-Holland, Amsterdam, 1971.

叙述了俄罗斯学派在这个领域的工作。

Robinson, A., *Introduction to Model Theory and to the Metamathematics of Algebra*, North-Holland, Amsterdam, 1963.

用标准的西方观点看逻辑在代数中的应用。

Robinson, A., *Nonstandard Analysis*, rev. ed., North-Holland, Amsterdam, 1974.

由该学科创始人所著的关于无穷小元素的计算和其他主题的权威教材。

Sacks, G., *Saturated Model Theory*, W. A. Benjamin, Reading, Mass., 1972.

以莫利-沃特(Morley-Vaught)传统对模型论进行了简明的处理。

五、证明论

Boolos, G., *The Unprovability of Consistency*, Cambridge University Press, Cambridge, England, 1979.

Boolos, G., *The Logic of Provability*, Cambridge University Press, Cambridge, England, 1993.

Clote, P. and Krajiček, J., *Arithmetic, Proof Theory, and Computational Complexity*, Oxford University Press, Oxford, 1993.

最近这些领域间联系的概要。

Dragalin, A. G., *Mathematical Intuitionism: Introduction to Proof Theory*, Translations of Mathematical

Monographs **67**, American Mathematical Society, Providence, 1988.

在假设读者不具备相关背景的前提下，给出了 Kripke 框架和克林的可实现性语义的一些相同之处。

Girard, J. -Y. , *Proof Theory and Logical Complexity*, vol. I, Bibliopolis, Naples, 1987.

对数学家而言，相当易懂的证明论观点。

Girard, J. -Y. , *Proofs and Types* (with appendices and tr. by Lafont, Y. and Taylor, P.) Cambridge University Press, Cambridge, England, 1989.

包含了吉拉尔(Girard)著名的二阶直觉主义逻辑的正规化定理，这也是多形程序设计语言的基础。

Pohlers, W. , *Proof Theory, An Introduction*, LNMS **1407**, Springer-Verlag, Berlin, 1989.

Prawitz, D. , *Natural Deduction*, Almqvist and Wiskell, Stockholm, 1965.

明确提出演绎是证明论的基本数学对象的观点。

Schutte, K. , *Proof Theory*(J. N. Crossley, tr.), Springer-Verlag, Berlin, 1977.

Takeuti, G. *Proof Theory*, 2nd ed. , North-Holland, Amsterdam, 1987.

Urquhart, A. , "The complexity of propositional proofs", *B. Symbolic Logic*, **1**, 425-465, 1995.

近期关于命题逻辑中证明复杂性工作的综述。

六、递归论

(另见列表第五节三)

Cutland, N. , *Computability: An Introduction to Recursive Function Theory*, Cambridge University Press, Cambridge, England, 1980.

包含借助于寄存器机所进行的可计算性研究。

Davis, M. , *Computability and Unsolvability*, McGraw Hill, 1958(reprinted Dover, New York, 1982).

以波斯特正则系统观点的讲解。

Davis, M. , Matijasevič, Ju. and Robinson, J. , "Hilbert's tenth problem, Diophantine equations: positive aspects of a negative solution" in Browder[1976, 1.2], 323-378.

Kleene, S. , "General recursive functions of natural numbers", *Math. Annalen*, **112**, 727-742, 1936 (reprinted in Davis [1965, 2.1]).

Kreisel, G. and Tait, W. , "Finite definability of number-theoretical functions and parametric completeness of equation calculi", *Zeit. Math. Log. Grund. Math.* , **7**, 28-38, 1961.

Lerman, M. , *Degrees of Unsolvability*, Springer-Verlag, Berlin, 1983.

一本从整体上论述图灵度的标准教材。

Minsky, M. L. , "Recursive unsolvability of Post's problem of tag and other topics in the theory of Turing machines", *Annals of Mathematics*, **74**, 437-454, 1961.

Odifreddi, G. , *Classical Recursion Theory*, I, North-Holland, Amsterdam, 1989, vol. II, 1997.

整数上的递归论，包含许多简化的证明和进一步的解释。

Rogers, H. , *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967 (Reprinted by MIT Press, Cambridge, Mass. , 1988).

许多年来最好的递归论方面的书，仍然是极好的出发点。

Shepherdson, J. and Sturgis, H. , "Computability of Recursive Functions", *J. ACM*, **10**, 217-255, 1963.

Shoenfield, J. R. , *Recursion Theory, Lecture Notes in Logic* **1**, Springer-Verlag, Berlin, 1993.

关于基础的简要介绍。

Smullyan, R. , *Theory of Formal Systems*, rev. ed. , Princeton University, Press, Princeton, 1961.

在 Horn 子句公理的基础上，对于字符串的递归可枚举集的发展。而很多计算机科学家还没充分认识到它们，常常在 PROLOG 语言中重复其中的结果。

Soare, R. I. , *Recursively Enumerable Sets and Degrees*, Springer-Verlag, Berlin, 1987.

这个领域目前的标准教材。

Tourlakis, G., *Computability*, Reston Publishing Co., Reston, Va., 1984.

也提到了寄存器机的发展。

Weihrauch, K., *Computability*, Springer-Verlag, Berlin, 1987.

七、范畴逻辑

Fourman, M. P. and Scott, D. S., "Sheaves and Logic", in *Applications of Sheaves*(Fourman, Mulvey, and Scott, eds.), LNMS **753**, Springer-Verlag, Berlin, 1979, 302-401.

清晰地解释了逻辑对层方法(logical versus sheaf methods)。

Goldblatt, R. I., *Topoi, the Categorical Analysis of Logic*, North-Holland, Amsterdam, 1979; rev. ed., 1984.

Lambek, J. and Scott, P. J., *Introduction to Higher Order Categorical Logic*, Cambridge University Press, Cambridge, England, 1986.

一本关于范畴论和高阶直觉主义逻辑之间联系的标准教材。

Makkai, M. and Reyes, G. E., *First Order Categorical Logic: Model-theoretical Methods in the Theory of Topoi and Related Categories*, Springer-Verlag, Berlin, 1977.

八、逻辑代数

Birkhoff, G., *Lattice Theory*, 3rd ed., American Mathematical Society Colloquium Pub. **25**, American Mathematical Society, Providence, 1967.

Dwinger, P. and Balbes, R., *Distributive Lattices*, University of Missouri Press, Columbus, Mo., 1974.

Halmos, P., *Lectures on Boolean Algebras*, D. Van Nostrand, Princeton, 1963 (reprinted, Springer-Verlag, Berlin, 1974).

Henkin, L., Monk, J. and Tarski, A., *Cylindric Algebras*, 2nd ed., North-Holland, Amsterdam, 1985.

Rasiowa, H. and Sikorski, R., *The Mathematics of Metamathematics*, Monographie Matematycze, **41**, PWN, Warsaw, 1963.

Sikorski, R., *Boolean Algebras*, 3nd ed., Springer-Verlag, Berlin, 1969.

第四节　直觉主义逻辑、模态逻辑和时态逻辑

(另见列表第五节六)

一、手册

Gabbay, D. and Guenthner, F., *Handbook of Philosophical Logic*, **4** vols., D. Reidel, Dordrecht, 1983-85.

这本书对非经典逻辑的许多分支做了有用的概括。

二、直觉主义逻辑和构成主义

Beeson, M. J., *Foundations of Constructive Mathematics*, Springer-Verlag, Berlin, 1985.

百科全书式的处理。

Bishop, E. and Bridges, D., *Constructive Analysis*, Springer-Verlag, Berlin, 1985.

毕晓普 1967 年著作的修订版，指出如何为近代泛函分析给出一种自然的构造性处理。

Bridges, D. and Richman, F., *Varieties of Constructive Mathematics*, Cambridge University Press, Cambridge, England, 1987.

构造性方法的入门书。

van Dalen, D., Intuitionistic logic, in *The Handbook of Philosophical Logic*, vol. Ⅲ, 225-339, D. Reidel, Dordrecht, 1986.

目前关于直觉主义逻辑数学的最好简介。

Dragalin, A. G., *Mathematical Intuitionism: Introduction to Proof Theory*, Translations of Mathematical Monographs **67**, American Mathematical Society, Providence, 1988.

在假设读者不具备相关背景的前提下，给出了 Kripke 框架和克林的可实现性语义的一些相同之处。

Dummett, M., *Elements of Intuitionism*, Clarendon Press, Oxford, 1977。

直觉主义背后哲学原理的完整陈述，以及对贝斯(Beth)和克里普克语义的阐述。

Fitting, M., *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel, Dordrecht, 1983.

多种表方法。

Gabbay, D. G., *Semantical Investigations in Heyting's Intuitionistic Logic*, D. Reidel, Dordrecht, 1981.

对 Kripke 框架语义的深入研究，包括多种理论的可判定性和不可判定性以及对结构的递归和递归可枚举表示的探讨。

Heyting, A., *Intuitionism, An Introduction*, 3rd ed., North-Holland, Amsterdam 1971.

标准讲解早期直觉主义的一个较新的版本。

Martin-Löf, P., *Intuitionistic Type Theories*, Bibliopolis, Naples, 1984.

谓词直觉主义逻辑类型理论。Constable 的 NUPRL 系统的基础[1986, 5.6]。

Nerode, A., "Some lectures on intuitionistic logic", in *Logic and Computer Science* (Lectures given at the 1st session of the C. I. M. E., Montecatini Terme, Italy, 1988), G. Odifreddi, ed., LNMS **1492**, Springer-Verlag, Berlin, 1990.

Nerode, A. and Remmel, J. B., A survey of r. e. substructures, in *Recursion Theory* (A. Nerode and R. Shore, eds.), Proc. Symposia in Pure Math., **42**, American Mathematical Society, Providence, 1985, 323-373.

这篇文章给出了递归代数方法的介绍。递归代数比直觉主义代数更完全地发展了经典的模拟，两者可通过递归可实现性联系起来。

Troelstra, A., *Principles of Intuitionism*, Springer-Verlag, Berlin, 1969.

会议报告。

Troelstra, A. and van Dalen, D., *Constructivism in Mathematics*, 2 vols., North-Holland, Amsterdam, 1988.

标准百科全书式处理，其中的材料在其他地方很难找到。

三、λ演算和组合逻辑

Barendregt, H., *The Lambda Calculus: Its Syntax and Semantics*, rev. ed., North-Holland, Amsterdam, 1984.

标准百科全书式的处理。

Gunter, C., *Semantics of Programming Languages*, MIT Press, Cambridge, 1992.

Hindley, J. R. and Seldin, J. P., *Introduction to Combinators and Lambda Calculus*, Cambridge University Press, Cambridge, England, 1986.

简短易读。

Revesz, G., *Lambda Calculus, Combinators, and Functional Programming*, Cambridge University Press, Cambridge, England, 1988.

Smullyan, R., *To Mock a Mockingbird and Other Logic Puzzles*, Knopf, NY, 1985.

斯莫里安的难题书之一。对组合逻辑中自引用的严格但非传统的处理。

四、模态逻辑

van Benthem, J., *Modal Logic and Classical Logic*, Bibliopolis, Naples, 1983.

一本高级教材。

van Benthem, J., *A Manual of Intensional Logic*, 2nd ed., CSLI, Stanford, 1988.

Chellas, B. F., *Modal Logic: An Introduction*, Cambridge University Press, 1980.

可能是计算机科学文献中被引用次数最多的教材。

Fitting, M., *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel, Dordrecht, 1983.

表方法的百科全书式处理。

Hughes, G. E. and Cresswell, M. J., *An Introduction to Modal Logic*, Methuen, London, 1968.

Hughes, G. E. and Cresswell, M. J., *A Companion to Modal Logic*, Methuen London, 1984.

不错的基础教材。

Goldblatt, R. I., *Mathematics of Modality*, CSLI Lecture Notes, **43**, Center for the Study of Language and Information, Stanford, 1993.

Lemmon, E. J., *An Introduction to Modal Logic*: *The "Lemmon Notes"* (K. Segerberg, ed.), American Philosophical Quarterly Monograph Series, **11**, Basil Blackwell, Oxford, 1977.

Linsky, L., ed., *Reference and Modality*, Oxford University Press, Oxford, 1971.

收集了模态逻辑早期的经典论文，包括克里普克的基础性工作，以及各种含有哲学观点的文章。

Nerode, A., "Lectures on modal logic", in *Logic*, *algebra*, *and computation* (International summer school directed by F. L. Bauer at Marktoberdorf, Germany, 1989) (F. L. Bauer, ed.), Springer-Verlag, Berlin, 1991.

Popkorn, S., *First Steps in Modal Logic*, Cambridge University Press, Cambridge, England, 1994.

五、时态逻辑

van Benthem, J., *The Logic of Time*, Reidel, Dordrecht, 1983.

van Benthem, J., *A Manual of Intensional Logic*, 2nd ed., CSLI, Stanford, 1988.

Gabbay, D. M., *Investigations in Modal and Tense Logics with Applications to Problems in Philosophy and Linguistics*, D. Reidel, Dordrecht, 1977.

Goldblatt, R., *Logics of Time and Computation*, 2nd ed., CSLI Lecture Notes, **7**, Center for the Study of Language and Information, Stanford, 1992.

Prior, A., *Past*, *Present*, *and Future*, Clarendon Press, Oxford, 1967.

Rescher, N. and Urquhart, A., *Temporal Logic*, Springer-Verlag, Berlin, 1971.

Shoham, Y., *Reasoning about Change*, MIT Press, Cambridge, Mass., 1988.

第五节 逻辑与计算

一、手册、原始资料和综述

Abramsky, S., Gabbay D. M. and Maibaum, T. E. S., eds., *Handbook of Logic in Computer Science*, vols. 1-4, Oxford University Press, Oxford, 1992-95.

一本重要的新作，包含许多主题的大量文章。

Barr, A. and Feigenbaum, E., eds., *The Handbook of Artificial Intelligence*, 3 vols., Heuristic Press, Stanford, 1981.

这些文章提供了一个关于这个领域的很好的概述，第3卷的许多条目与这里包含的主题是直接相关的。

Bledsoe, W. and Loveland, D., *Automatic Theorem Proving after 25 years*, American Mathematical Society, Providence, 1984.

Boyer, R. S. and Moore, J. S., eds., *A Computational Logic Handbook*, Academic Press, Boston, 1988.

主要面向自动化定理证明。

Gabbay, D. M., Hogger, C. J. and Robinson, J. A., *Handbook of Logic in Artificial Intelligence and Logic Programming*, Oxford University Press, Oxford 1993-.

Lassez, J.-L. and Plotkin, G., eds., *Computational Logic*: *Essays in Honor of Alan Robinson*, MIT Press, Cambridge, Mass., 1991.

这本书包含了计算逻辑主要方面的许多综述以及当前的研究论文。

van Leeuwen J., ed., *Handbook of Theoretical Computer Science*, 2 vols., North-Holland, Amsterdam and MIT Press, Cambridge, Mass., 1990.

集体成果。总结了当时理论计算机科学的状况，特别是逻辑的许多应用，例如程序的逻辑。

Nerode, A., "Applied logic", in *The Merging of Disciplines*: *New Directions in Pure*, *Applied*, *and Computational Mathematics* (R. E. Ewing, K. I. Gross and C. F. Martin, eds.), Springer-Verlag, Berlin,

127-163.

1986 年之前逻辑在计算机科学中某些应用的简略介绍。

Siekmann, J. and Wrightson, G., eds., *Automation of Reasoning*, 1957-70, 2 vols., Springer-Verlag, Berlin, 1983.

本卷包含了大部分自动推理早期的论文，许多都可作为不错的阅读材料。

二、综合教材

Börger, E. *Computability*, *Complexity*, *Logic*, North-Holland, Amsterdam, 1989.

Davis, M., Sigal, R. and Weyuker, E., *Computability*, *Complexity*, *and Languages*: *Fundamentals of Theoretical Computer Science*, Academic Press, San Diego, 1994.

Davis, M. and Weyuker, E., *Computability*, *Complexity*, *and Languages*, Academic Press, New York, 1983.

Davis, R. E., *Truth*, *Deduction and Computation*, Computer Science Press, N. Y., 1989.

Fitting, M., *Computability Theory*, *Semantics*, *and Logic Programming*, Oxford University Press, Oxford, 1987.

Gallier, J., *Logic for Computer Science*, *Foundations of Automatic Theorem Proving*, Harper and Row, New York, 1986.

基于矢列式和消解，还包括截消去、Hauptsatz、SLD-消解以及许多种类的逻辑。

Harel, David, *Algorithmics*: *The Spirit of Computing*, Addison-Wesley, Reading, Mass., 1987.

Lewis, H. and Papadimitriou, C., *Elements of the Theory of Computation*, Prentice Hall, Englewood Cliffs, NJ, 1981.

Manna, Z., *Mathematical Theory of Computation*, McGraw-Hill, New York, 1974.

Manna, Z. and Waldinger, R., *The Logical Basis for Computer Programming*, Addison-Wesley, Reading, Mass., 1985.

Robinson, J. A., *Logic*: *Form and Function*, North-Holland, Amsterdam, 1979.

Schöning, U., *Logic for Computer Scientists*, Birkhauser, Boston, 1994.

Sperschneider, V. and Antoniou, G., *Logic*: *A Foundation for Computer Science*, Addison-Wesley, Reading, Mass., 1991.

三、复杂性理论

Balcazar, J. L., Diaz, J. and Gabarro, J., *Structural Complexity*, 2 vols., Springer-Verlag, Berlin, 1988, 1990.

目前介绍复杂性理论的一本好书。

Clote, P. and Krajiček, J., *Arithmetic*, *Proof Theory*, *and Computational Complexity*, Oxford University Press, Oxford, 1993.

关于近期这些领域之间联系的梗概。

Garey, M. R. and Johnson, D. S., *Computers and Intractability*: *A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, New York, 1979.

仍然是学习 P 和 NP 问题的最好材料。

Hartmanis, J., ed., *Computational Complexity Theory*, Proc. Symp. App. Math., **38**, American Mathematical Society, Providence, 1989.

来自美国数学学会短期课程的讲义。

Hopcroft, J. E. and Ullman, J. D., *Introduction to Automata Theory*, *Languages and Computation*, Addison-Wesley, Reading, Mass., 1979.

为计算机系学生写的关于图灵机、自动机和基本复杂性的标准教材。

Machtey, M. and Young, P., *An Introduction to the General Theory of Algorithms*, North-Holland, Amsterdam,

1978.

Papadimitriou, C. H., *Computational Complexity*, Addison-Wesley, Reading, Mass., 1994.

Statman, R., "Intuitionistic propositional logic is polynomial space complete", *Theoretical Computer Science*, **9**, 67-72, 1979.

Urquhart, A., "The complexity of propositional proofs", *B. Symbolic Logic*, **1**, 425-465, 1995.

关于命题逻辑中证明复杂性近期工作的一篇综述。

四、逻辑式程序设计和 PROLOG

Apt, K. R., "Ten Years of Hoare's Logic: A Survey——Part I", *ACM TOPLAS* **3**, 431-483, 1981.

Apt, K. R., "Introduction to logic programming", in van Leeuwen [1990, 5.1] vol. B, 493-574, 1990.

Apt, K. R., *From Logic Programming to* PROLOG, to appear, 1996.

Apt, K. R., Blair, H. A. and Walker, A., "Towards a theory of declarative knowledge", in *Foundations of Deductive databases and Logic Programming*(J. Minker, ed.), Morgan Kaufmann, Los Altos, Ca., 1988.

Apt, K. R. and Pedreschi, D., "Studies in Pure Prolog", in *Computational Logic*(J. Lloyd, ed.), Springer-Verlag, Berlin, 1991.

Bezem, M., "Characterizing termination of logic programs with level mappings", *Proc. North American Logic Programming Conference*(E. L. Lusk and R. A. Overbeek, eds.), vol. 1, 69-80, MIT Press, Cambridge, Mass., 1989.

Bratko, I., *PROLOG Programming for Artificial Intelligence*, Addison-Wesley, Reading, Mass., 2nd ed., 1990.

Boizumault, P., *The Implementation of Prolog*, Princeton University Press, Princeton, N. J., 1993.

法国学者在 PROLOG 的发展中起了很大作用。这是来自法语的译本。

Clark, K. L., "Negation as failure", in *Logic and Databases* (H. Gallaire and J. Minker, eds.), 293-322, Plenum Press, New York, 1978.

Clocksin, W. F. and Mellish, C. S., *Programming in PROLOG*, 3rd revised and extended edition, Springer-Verlag, Berlin, 1987.

Dodd, T., PROLOG, *A Logical Approach*, Oxford University Press, Oxford, 1989.

Doets, K., *From Logic to Logic Programming*, MIT Press, Cambridge, Mass., 1994.

Gelfond, M. and Lifschitz, V., "Stable semantics for logic programming", in *Logic Programming*, *Proc.* 5th *International Conference and Symposium on Logic Programming* (Seattle, 1985) (R. Kowalski and K. Bowen, eds.), MIT Press, Cambridge, Mass., 1988.

Kowalski, R., *Logic for Problem Solving*, North-Holland, Amsterdam, 1979.

Lloyd, J. W., *Foundations of Logic Programming*, 2nd extended edition, Springer-Verlag, Berlin, 1987.

这个领域的标准介绍，且包含一个丰富的参考书目。

Maier, D. and Warren, D., *Computing with Logic*: *Logic Programming with* PROLOG, Benjamin/Cummings, Menlo Park, Ca., 1988.

结合 PROLOG 研究有关消解的逻辑，与数据库有极强的联系。

Martelli, A. and Montannari, U., "An efficient unification algorithm", *ACM Transactions on Programming Languages and Systems*, **4**, 258-282, 1982.

Mizoguchi, F., *PROLOG and its Applications*: *A Japanese Perspective*, Chapman and Hall, London, 1991.

PROLOG 在日本有着广泛的应用。本书译自其日文版。

O'Keefe, R. A., *The Craft of PROLOG*, MIT Press, Cambridge, Mass., 1990.

Robinson, J. A., "Logic and Logic Programming", *Communications of the ACM*, **35**, 40-65, 1992.

由该领域开创者之一撰写的简史。

Shepherdson, J. C., "Logics for negation as failure", in *Logic from Computer Science*(Y. N. Moschovakis, ed.), 521-583, MSRI Publications **21**, Springer-Verlag, Berlin, 1992.

Sterling, L., *The Practice of PROLOG*, MIT Press, Cambridge, Mass., 1990.

Sterling, L. and Shapiro, E., *The Art of Prolog: Advanced Programming Techniques*, MIT Press, Cambridge, Mass., 1986.

Thayse, A., ed., *From Standard Logic to Logic Programming*, Wiley, Chichester, 1988.

涉及人工智能的动机。参阅 Thayse[1989, 5.6]。

五、非单调逻辑

Besnard, P., *An Introduction to Default Logic*, Springer-Verlag, Berlin, 1989.

Brewka, G. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*, Cambridge University Press, Cambridge, England, 1991.

Doyle, J., "A truth maintenance system", *Artificial Intelligence Journal*, **12**, 231-272, 1979.

Ginsberg, M., ed., *Readings in Nonmonotonic Reasoning*, Morgan and Kaufmann, Los Altos, Ca., 1987.

Hintikka, J., *Knowledge and Belief*, Cornell University Press, Ithaca, N. Y., 1962.

McCarthy, J., "Circumspection—a form of nonmonotonic reasoning", *Artificial Intelligence*, **13**, 27-39, 1980.

Marek, W., Nerode, A. and Remmel, J. B., "Non-Monotonic rule systems" I, II, *Mathematics and Artificial Intelligence*, **1**, 241-273, 1990, **5**, 229-263, 1993.

Marek, W. and Truszczynski, M., *Nonmonotonic Logic: Context-dependent Reasoning*, Springer-Verlag, Berlin, 1993.

第一本真正论述该学科的数学方面内容的教材。

Minsky, M., "A framework for representing knowledge", in *The Psychology of Computer Vision* (P. Winston, ed.), 211-272, McGraw-Hill, New York, 1975.

Moore, R. C., "Possible world semantics for autoepistemic logic", *in Proceedings 1984 Nonmonotonic Reasoning Workshop* (AAAI), New Paltz, N. Y., 1984.

Moore, R. C., "Semantical considerations on nonmonotonic logic", *Artificial Intelligence*, **25** (1), 75-94, 1985.

Parikh, R., ed., *Theoretical Aspects of Reasoning about Knowledge* (*TARK 90*), Morgan Kaufmann, Los Altos, Ca., 1990.

Reiter, R., "A logic for default reasoning", *Artificial Intelligence*, **13**, 81-132, 1980.

Reiter, R., "Nonmonotonic reasoning", *Ann. Rev. Comp. Sci.*, **2**, 147-186, 1980.

Vardi, M., ed., *Theoretical Aspects of Reasoning about Knowledge* (*TARK 88*), Morgan and Kaufmann, Los Altos, Ca., 1988.

六、直觉主义，模态和时态逻辑

(另见列表第四节一～五)

Constable, R., *Implementing Mathematics with the* NUPRL *Proof Development System*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

作为数学工作者的助手，NUPRL 实现了 Martin-Löf 谓词直觉主义类型理论。

Galton, A., *Temporal Logics and their Applications*, Academic Press, London, 1987.

Goldblatt, R. I., *Axiomatizing the Logic of Computer Programming*, LNCS **130**, Springer-Verlag, Berlin, 1982.

Goldblatt, R. I., *Logics of Time and Computation*, 2nd ed., CSLI Lecture Notes, **7**, Center for the Study of Language and Information, Stanford, 1992.

Halpern, J. Y. and Moses, Y. O., "A guide to modal logics of knowledge and belief", *Proc. 9th Int. Joint Conf. on Artificial Intelligence* (IJACI).

Manna, Z. and Waldinger, R., *The Logical Basis of Computer Programming*, Addison-Wesley, Reading, Mass., 1985.

Smets, P., Mamandi, E. H., Dubois, D. and Padre, H., *Nonstandard Logics for Automated Reasoning*, Academic Press, London, 1988.

Thayse, A., ed., *From Modal Logic to Deductive Databases*, Wiley, Chichester, 1989.

令人印象深刻地介绍了各类模态逻辑，模态逻辑是人工智能工作的基础。参阅 Thayse[1958, 5.4]。

Thayse, A., ed., *From Natural Language Processing to Logic for Expert Systems*, Wiley, Chichester, 1991.

广泛地论述了逻辑方法在人工智能多个领域中的应用。

Thistlewhite, P. B., McRobbie, M. A. and Meyer, R. A., *Automated Theorem-proving in Nonclassical Logics*, Pitman, London, 1988.

Turner, R., *Logics for Artificial Intelligence*, Halsted Press, Chichester, 1984.

Wallen, L., *Automated Proof Search in Nonclassical Logics*, MIT Press, Cambridge, Mass, 1990.

这本书和 Thistlewhite[1988, 5.6]探索了怎样将各种类型的非经典逻辑自动化。

七、自动演绎和程序验证

(另见列表第五节六)

de Bakker, J., *Mathematical Theory of Program Correctness*, Prentice-Hall, Englewood Cliffs, N. J., 1980.

Bibel, W., *Automated Theorem Proving*, Vieweg, Braunschweig, 2nd rev. ed., 1987.

Bledsoe, W. and Loveland, D., *Automatic Theorem Proving after 25 years*, American Mathematical Society, Providence, 1984.

Boyer, R. S. and Moore, J. S., *A Computational Logic*, Academic Press, New York, 1979.

介绍了一种自动逻辑，试图在基础数论中找出正确的归纳证明，为此做了很多探索。它是程序开发和验证的研究基础。

Boyer, R. S. and Moore, J. S., eds., *The Correctness Problem in Computer Science*, Academic Press, New York, 1981.

Burstall, R., "Some techniques for proving correctness of programs which alter data structures", *Machine Intelligence*, **7**, 23-50, 1972.

Chang, C.-L. and Lee, C.-T., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

采用厄布朗域的观点来讲述基础逻辑。

Colburn, T. R., Fetzer, J. H. and Rankin, T. L., eds., *Program Verification: Fundamental Issues in Computer Science*, Kluwer Academic Press, Dordrecht, Holland 1993.

Davis, M., "The prehistory and early history of automated deduction", in Siekmann and Wrightson [1983, 5.1], 1-28.

对该领域早期历史的很好的浏览。

Davis, M. and Putnam, H., "A computing procedure for quantification theory", *J. ACM*, **7**, 201-216, 1960 (reprinted in Siekmann and Wrightson [1983, 5.1], vol. 1, 125-139).

Floyd, R., "Assigning meaning to programs", *Proceedings of Symposia in Applied Mathematics*, **19**, 19-32, American Mathematical Society, Providence, 1967.

Gries, D., *The Science of Programming*, Springer-Verlag, New York, 1981.

Harel, D., *First Order Dynamic Logic*, Springer-Verlag, Berlin, 1979.

Hoare, C., "An axiomatic basis for computer programming", *Communications of the ACM*, **12**, 576-583, 1969.

Hoare, C. and Shepherdson, J., eds., *Mathematical Logic and Programming Languages*, Prentice-Hall, Englewood Cliffs, N. J., 1985.

Huet, G. and Plotkin, G., eds., *Logical Frameworks*, Cambridge University Press, Cambridge, England, 1991.

Loeckx, J. and Sieber, K., *The Foundations of Program Verification*, 2nd ed., Wiley, New York, 1987.

Loveland, D. , *Automated Theorem Proving: A Logical Basis*, North-Holland, Amsterdam, 1978.

各种消解的完整展示。

Ramsay, A. , *Formal Methods in Artificial Intelligence*, Cambridge Tracts in Theoretical Computer Science **6**, Cambridge University Press, Cambridge, England, 1988.

一本内容广泛的高级教材，处理许多与人工智能相关的逻辑方法。

Robinson, J. A. , "A Machine oriented logic based on the resolution principle", *J. ACM*, **12**, 23-41, 1965 (reprinted in Siekmann and Wrightson [1983, 2. 1]).

Turing, A. M. , "Checking a large routine", in *Report of a Conference on High Speed Automatic Calculating Machines*, University Mathematical Library, Cambridge, England, 1949.

Wos, L. , Overbeek, R. , Lusk, E. and Boyle, J. , *Automated Reasoning*, Prentice-Hall, Englewood Cliffs, N. J. , 1984.